우리식조작체계문고

동적 웨브페지작성법

ASP
JSP
PHP
STYLE SHEET

교육성교육정보쎈터
주체97(2008)년

차 례

머 리 말

위대한 령도자 **김정일**동지께서는 다음과 같이 지적하시였다.

《정보산업에 대한 인식을 바로 가지고 정보산업시대의 요구에 맞게 일하여야 합니다.》(《**김정일**선집》 제15권, 195페지)

그 어느때보다도 급속히 발전하고있는 오늘의 과학과 기술은 콤퓨터와 콤퓨터망을 떠나서는 생각할수 없다. 콤퓨터망을 효과적으로 리용하면 과학기술을 발전시켜 정치, 경제, 문화 등 사회생활의 모든 분야에서 제기되는 문제들을 신속히 편리하게 높은 수준에서 해결해나갈수 있다.

콤퓨터망 사용자들은 HTML언어를 비롯한 각이한 환경들에서 리용할수 있는 동적웨브페지작성언어 등 웨브페지작성에 기초로 되는 지식을 잘 알고있어야 망가입자로서의 주동적이며 능동적인 활동을 실리적으로 할수 있다.

이 책은 웨브페지를 만드는데 필요한 HTML의 꼬리표들에 대하여서와 동적웨브페지작성언어들인 ASP와 JSP, PHP에 대하여 대비적으로 학습할수 있도록 내용을 구성하였다.

현 시대의 핵심기초기술의 하나인 정보기술을 급속히 발전시키며 가까운 앞날에 전국적인 광역망을 구축하여 인민경제의 현대화, 정보화를 힘있게 다그치는데 필요한 정보기술전문가들을 키우는 사업에 이 책이 도움이 되리라고 본다.

우리는 강성대국건설에서 과학기술을 중시할데 대한 당의 의도에 맞게 나라의 정보기술발전을 위하여 꾸준히 노력하여야 할것이다.

제 1 장. HTML 언어

제1절. HTML의 기초

1.1.1. HTML 의 개념

HTML은 Hyper Text Markup Language의 략자로서 웨브열람기에서 실행되는 하이퍼본문기능을 가진 문서를 만드는 언어이다. 여기서 하이퍼본문기능이라는것은 웨브문서의 임의의 위치에서 해당 부분(알려고 하는 정보 혹은 내용물, 웨브거점주소 등)을 선택했을 때 다른 곳(웨브문서, 웨브거점)으로 이동할수 있도록 련결해주는 기능을 말한다.

HTML에서는 련결시키려는 하이퍼본문이 기본구조로 된다. HTML문서는 파일확장자가 *.html 또는 *.htm이다.

일반문서를 작성할 때 Micrsoft word를 사용하듯이 HTML을 작성할 때에도 일정한 도구들이 필요하다. 즉 HTML을 입력하기 위한 일종의 편집기가 있어야 한다.

아주 초보적인 편집기는 Notepad인데 HTML문법을 모두 알고 작성해야 한다는 어려움이 있지만 Notepad를 학습하는것은 기초를 튼튼히 하고 언제 어디서나 HTML문서를 만들수 있게 한다는 점에서 아주 유리하다. 한편 HTML문법을 모르고도 좀 더 간편하게 HTML을 만들수 있게 하는 Front Page, Hot Dog, Namo web editor, Dreamweaver 등 많은 HTML전용편집기들이 있다.

1.1.2. 꼬리표의 개념

꼬리표란 HTML을 작성하기 위한 하나의 재료로서 여러개의 꼬리표가 모여서 하나의 완성된 HTML을 이루게 된다.

꼬리표의 형식은 다음과 같다.

```
<꼬리표이름 선택항목 1=선택항목 1 값
            선택항목 2=선택항목 2 값 …>
```

- 꼬리표이름

  꼬리표의 종류는 무수히 많다. 어떤 꼬리표는 글자를 전문적으로 담당하고 어떤 꼬리표는 그림을 담당하고 어떤 꼬리표는 움직임을 담당하는 등 여러가지 역할을 수행하는 꼬리표들이 많다. 이러한 매 꼬리표들에는 이름이 붙는다. 례를 들어 서체꼬리표를 써넣으려고 한다면 꼬리표이름으로 font를 써주면 된다

- 선택항목=선택항목값(생략가능)

매개 꼬리표에는 선택항목이 있다. 례를 들어 글자를 담당하는 서체꼬리표라면 글자를 얼마만큼 크거나 작게 하겠는가, 색갈은 어떤것으로 하겠는가, 서체는 어떤것으로 할것인가 등의 선택항목이 있다.

매 꼬리표에 따르는 선택항목들은 서로 다르다. 만일 font꼬리표를 사용하려고 한다면 font꼬리표에는 color라는 선택항목이 있다. 실례로 color의 값을 red로 주는 꼬리표문장형식은 <font color=red>와 같다.

- 꼬리표의 완료

무엇을 하든지 끝마무리가 중요한것만큼 모든 꼬리표들은 시작꼬리표와 함께 끝꼬리표를 가지고있다. 끝꼬리표는 시작꼬리표의 이름앞에 사선기호(/)를 붙인다. 즉 형식은 </꼬리표이름>이다.

- 대소문자구별

대소문자는 구별하지 않아도 된다.

1.1.3. HTML의 기본형식

HTML의 기본형식은 다음과 같다.

```
<html>
<head>
<title>제목</title>
</head>
<body>
내용
</body>
</html>
```

구체적으로 설명하면 아래와 같다.

- <html>, </html>

<html>은 《지금부터 HTML문서를 시작하겠다.》라는 뜻이다. 이것은 가장 기본적인 꼬리표이므로 문서에 제일 먼저 입력해주어야 한다. </html>은 HTML문서의 끝을 의미하므로 제일 마지막에 입력해주어야 한다.

- <head>, </head>

<head>는 말그대로 문서의 머리말을 입력하는 부분이다. 따라서 문서의 대략적인 특성이나 정보를 <head>와 </head>사이에 입력한다.

- <title>, </title>

<title>은 문서의 제목을 표시하는 꼬리표이다. 문서를 요약시킬만한 제목을 넣는

꼬리표이므로 제목꼬리표는 <head>꼬리표안에 넣는다. 문서의 제목은 <title>과 </title>사이에 넣어준다.

- <body>, </body>

HTML에서 <body>는 그 문서의 기본내용이 서술되는 부분이다. 즉 <body>란 《지금부터 본격적인 내용을 시작하겠다.》라는 뜻이다. 그 내용은 <body>와 </body>안에 입력하면 된다.

<body>의 내용을 어떻게 작성하는가에 따라 화면이 달라진다. 따라서 모든 꼬리표들은 <body></body>안에 잘 조합하여 넣어야 한다.

제2절. 본문형식화

1.2.1. 서체꼬리표(font)

화면에 출력하는 글자의 크기를 변화시킬 때 사용하는 꼬리표는 <hn>이다.
형식은 다음과 같다.

```
<hn> hn의 실례입니다</hn>
```

실 례:

h1의 실례입니다
h2의 실례입니다
h3의 실례입니다

여기서 n은 1부터 6까지인데 n값이 클수록 현시되는 글자가 작아진다.

다음으로 많이 쓰이는것이 font인데 그 형식은 다음과 같다.

```
<font size=글자의 크기 color=색값 face=서체이름>내용</font>
```

마지막에 로 완료하지 않으면 계속 그 글형식이 적용된다.

실 례:

내용

색값은 우의 실례와 같이 영문으로 써도 되고 "#ffffff"와 같이 써도 된다.
또는 여기에 marquee를 섞어 쓰기도 하는데 그렇게 하면 움직이는 효과를

나타낸다.

실 례:

<marquee>내용</marquee>

1.2.2. 본문서식끄리표

- <p>

형식은 다음과 같다.

내용

본문에서 단락을 구분할 때 사용하는 끄리표로서 자동적으로 행바꾸기를 한 다음 한 행을 비여 놓고 다음 행에서 시작한다. 여기에 align이라는 선택항목을 줄수 있는데 이것은 정렬방식을 나타낸다.

실 례:

```
<p align="center">
조선을 위하여 배우자!</p>
```

이것을 실행하면 한 행을 비우면서 본문내용을 중심맞추기하여 현시한다.

align선택항목은 <p>끄리표만 아니라 div, marquee, hr, h1, td, image를 비롯한 많은 끄리표들에도 쓰인다.

- <div>

형식은 다음과 같다.

내용

이 끄리표는 align속성값에 따라 문장의 표시위치를 설정한다.

《속성값》에는 left, center, right을 입력할수 있다. 그러면 문장내용이 왼쪽, 중심, 오른쪽맞추기된다.

**-
**

형식은 다음과 같다.

내용

문서에서 줄바꾸기를 진행할 때 사용한다. Enter건의 역할과 같다고 리해하면 된다. 끄리표가 적용되지 않는 문서에는 br를 쓸 필요없이 Enter건으로 처리해도 된다.

- <center>

형식은 다음과 같다.

<center>내용</center>

글을 문서의 중심에 배치할 때 쓰인다. 이것은 <p align= center>현시내용</ p>와 같은 효과를 낸다.

- <hr>

형식은 다음과 같다.

<hr>

이 꼬리표는 여러가지 모양의 가로선을 그릴 때 사용한다. <hr>에는 다음과 같은 선택항목들이 있다.

size : 가로선의 두께를 지정
width: 가로선의 크기를 지정
color : 색을 지정
align : 가로선의 위치를 지정

실 례:

<hr color=red size=50>

이렇게 주면 길이가 50인 빨간색의 가로선이 생긴다.

- <big>

형식은 다음과 같다.

<big>현시내용</big>

서체를 기본글형식보다 한 단계 크게 나타낼 때 사용한다.

<big>꼬리표와 반대되는 기능을 수행하는 꼬리표가 있는데 그것이 바로 <small>꼬리표이다.

1.2.3. 글자서식꼬리표

-

형식은 다음과 같다.

현시내용

글자를 진하게 표시한다.

- <i>

형식은 다음과 같다.

<i>현시내용</i>

글자를 경사체로 만든다.

- <u>

형식은 다음과 같다.

<u>내용</u>

글자에 밑줄을 그어준다.

- <blink>

형식은 다음과 같다.

<blink>내용</blink>

글자를 깜빡이게 한다.

-

형식은 다음과 같다.

 내용

글자를 진하게 표시한다.

- <strike>

형식은 다음과 같다.

<strike> 내용 </strike>

글자중간에 줄을 긋기한다.

- <sub>

형식은 다음과 같다.

_{내용}

글을 아래첨자로 표시하려고 할 때 리용한다.

- <sup>

형식은 다음과 같다.

^{내용}

글을 웃첨자로 보여준다.

1.2.4. 입력담당끄리표(input)

입력끄리표는 단추나 본문칸을 만드는 끄리표이다.

형식은 다음과 같다.

<input type="형식" value="현시내용">

형식에는 submit, button, radio, checkbox, text 등이 속한다.

실례를 들어보면

<input type=submit value="현시내용">

<input type=button value="현시내용">

button과 submit는 완전히 같지 않은데 button은 단순히 모양만을 만드는것이고 submit는 거기에 질문전송의 기능을 더 가지고있다. 따라서 <form>끄리표와 함께 쓸 때에는 submit를 쓴다.

radio는 단일선택단추를 만들어 주는 형식이다.

checkbox는 다중선택단추를 만들어 주는 형식이다.

text는 본문칸을 만드는 형식이다.

text형식에는 size선택항목을 추가할수 있다.

실 례:

<input type=text size=10 value="현시내용">:

maxlength라고 최대입력가능선택항목이 있는데 빈칸에 최대로 입력이 가능한 문자의 개수를 지정한다. 만일 maxlength=4로 하는 경우에 네글자를 초과하면 자료입력이 되지 않는다.

input로 단추나 본문칸을 만들었다고 해도 찰칵하거나 문자를 입력하면 어떤 변화가 생기는것은 아니다. input는 단순히 그런 모양만 만들어주는 역할을 한다.

input와 자주 리용되는것중의 하나가 <form>인데 주로 CGI나 어떤 특정한 주소로 어떤 내용을 보내게 하는 역할을 한다.

1.2.5. 운동담당끄리표(marquee)

marquee는 글자나 여러 객체를 움직이게 하는 효과를 내는데 marquee를 잘 리용하면 글자가 여러가지 형태로 움직이도록 할수 있다.

물론 3차원적인 운동이나 기타 복잡한 불규칙운동은 DHTML언어에 대하여 알아야

실현할수 있지만 marquee로도 어느정도 표현은 할수 있다.

형식은 다음과 같다.

<marquee> 현시내용 </marquee>

</marquee>는 《해당한 꼬리표를 끝내라.》는 뜻이다. 이 끝꼬리표를 리용하면 <marquee>와 </marquee>사이에 있는 현시내용만이 움직인다.

이 꼬리표의 선택항목들에는 배경색(bgcolor), 운동방향(direction), 왕복운동(behavior), 이동시간(scrolldelay), 이동량(scrollamount), 이동너비(width), 이동높이(height), 반복회수(loop)들이 있다.

실 례:

<marquee> 현시내용 </marquee>//우측에서 좌로 이동

<marquee bgcolor=red>글자입력</marquee>//배경색은 붉은색

<marquee direction="down">글자입력</marquee>//운동방향은 아래로(인용부호는 생략할수도 있다.) 만약 up으로 하면 우로 올라가며 right로 하면 좌측에서 우측으로 이동한다.

<marquee behavior=alternate>현시내용</marquee>//왕복운동

<marquee scrolldelay=200>현시내용</marquee>//이동시간이 200인데 수자가 클수록 지체시간이 길어진다. 즉 움직이는 속도가 떠진다.

<marquee scrollamount=1>현시내용<marquee>//움직이는 이동량이 1인데 수자가 작을수록 세밀하게 움직인다.

<marquee width=60>현시내용</marquee>//움직이는 너비가 60이다.

<marquee direction=up height=40>현시내용</marquee>//움직이는 높이가 40이다.

<marquee loop=50>현시내용</marquee>//반복회수가 50이다.

띠의 속도와 글자의 속도를 다르게 하려는 경우에는 2중 marquee를 리용해야 한다.

실 례:

<marquee><marquee bgcolor=pink>현시내용</marquee></marquee>

bgcolor는 움직이는 글자에 대한 배경색을 의미하는데 <marquee>꼬리표를 하나만 주었을 때 배경부분은 정지되고 글자만 움직인다. 여기에 <marquee>를 더 추가하면 bgcolor가 적용된 배경부분도 움직이게 되고 원래 움직이던 글자도 marquee의 효과를 더 받아 더 빨리 움직이게 된다. 물론 글자는 배경부분안에서 움직이고 있었으므로 그 령역을 벗어나지 않고 그안에서 더욱 빨리 움직이게 된다.

<marquee>꼬리표를 리용하여 글자가 깜빡이는 효과를 줄수 있는데 그것은 좌우공

간을 정해주는 선택항목인 width와 이동량을 정해주는 선택항목인 scrollamount를 리용하여 실현할수 있다. 주어진 공간안에서 이동량이 다르면 표현되는 운동도 차이가 난다. 따라서 같은 scrollamount값을 주더라도 width가 작을수록 더욱 깜빡거리는 효과가 난다.

실 례:

<marquee width=100 scrollamount=80>깜빡깜빡</marquee>

<marquee width=350 scrollamount=80>깜빡깜빡</marquee>

만일 width의 값은 같고 scrollamount를 다르게 준다면

즉

<marquee width=350 scrollamount=300>이것은 300</marquee>

<marquee width=350 scrollamount=2>이것은 2</marquee>

이라면 scrollamount의 값에 따라 그 움직임도 차이가 나게 된다. 이 값을 너무 크게 주면 보이지 않으며 작게 주면 깜빡이는게 아니라 부드럽게 움직이게 된다.

실 례:

<marquee width=200 scrollamount=70>★</marquee>

<marquee width=150 scrollamount=47>★</marquee>

<marquee width=200 scrollamount=48>☆</marquee>

<marquee width=100 scrollamount=45>☆</marquee>

<marquee width=130 scrollamount=57>☆</marquee>

<marquee width=180 scrollamount=60>★</marquee>

<marquee width=200 scrollamount=68>☆</marquee>

<marquee width=143 scrollamount=47>☆</marquee>

<marquee width=150 scrollamount=52>☆</marquee>

<marquee width=170 scrollamount=58><font

color=orange>★</marquee>
<marquee width=130 scrollamount=57>☆</marquee>
<marquee width=180 scrollamount=60>★</marquee>
<marquee width=200 scrollamount=68>☆</marquee>
<marquee width=200 scrollamount=70>☆</marquee>
<marquee width=150 scrollamount=47>★</marquee>
<marquee width=200 scrollamount=48>☆</marquee>

marquee는 단지 Internet Explorer에서만 지원되고 Netscape에서는 되지 않는다.

1.2.6. 그림담당꼬리표(img)

이 꼬리표는 그림을 보여주는 역할을 하는 꼬리표로서 그림의 주소만 알고 있으면 쉽게 실현할수 있다.

형식은 다음과 같다.

<img src= "그림주소" >

례를 들어 그림파일 《kitty.gif》의 경로가 《D:\image\gif》이라면

< img src= "D:\image\gif\kitty.gif>"

와 같은 형식으로 입력한다.

여기에 marquee라든가 여러 다른 꼬리표를 삽입해서 여러가지 기능을 실현할수 있다.

실 례:

<marquee direction=up scrollamount=3 behavior=alternate height=130>
<marquee scrollamount=2 direction=right><img src=kitty.gif>
</marquee></marquee>

그림의 크기도 조절할수 있는데 그것은 width와 height를 가지고 조절하면 된다. 가로세로의 비률을 같은 비률로 하고 크기만 조절하려면 width를 조절하고 모양자체를 바꾸게 하려면 두 값을 다 변경해야 한다.

실 례:

<img width=" 20" height=" 20" src= "D:\image\gif\kitty.gif>"

다음으로 그림과 글자를 함께 현시할 때 글자의 위치를 정하는 방법을 고찰한다.

이것은 Align선택항목을 리용하여 실현할수 있다.

실 례:

<img align=left src=kitty.gif>현시내용

우와 같은 코드를 실행하면 그림은 왼쪽, 글자는 오른쪽우로부터 정렬된다. 만일 글자를 오른쪽 아래로 정렬하고싶으면
꼬리표를 리용한다.

실 례:

<img align=left src=kitty.gif>

현시내용

를 추가하는것은 "align=middle"을 주는것보다 좀 더 세밀하게 위치를 지정해 줄수 있다.

마찬가지로 "align=right"하면 그림은 오른쪽, 글자는 왼쪽에 배치된다.

정렬방식에는 left, right, top, middle, bottom, baseline, texttop, absmiddle, absbottom 등이 있다.

1.2.7. 음악담당꼬리표(bgsound)

음악연주기능을 수행하는 꼬리표이다.

형식은 다음과 같다.

<bgsound src="음악 주소">

실 례:

<bgsound src=http://www.kebi.com/kck2002/3.wav>

<bgsound>꼬리표도 선택항목을 가지고있다.

음악을 한번 연주하려면 loop명령어를 다음과 같이 주면 된다.

실 례:

<bgsound src=http://ww.kebi.com/kck2002/3.wav loop=1>

<embed>를 사용하여 중지시킬수도 있고 autoplay를 리용하여 자동적으로 연주되도록 할수도 있다. 즉

<embed src=주소 width=너비수자 height=높이수자hidden=true/false autoplay=true/false loop=반복회수>

이다.

1.2.8. 련결꼬리표

어떤 본문이나 그림을 찰칵하면 그에 해당한 다른 자료들을 련달아 찾을수 있게 하는 꼬리표이다. 이것에는 두가지가 있다.

하나는 <a>라는 꼬리표를 리용한 방법이고 다른 하나는 <form>+<input>를 리용한

방법이다.

<a>끄리표의 기본형식은 다음과 같다.

```
<a href="이동하려고 하는 주소" target="프레임이름">현시내용</a>
```

"frame이름" 에는 목적프레임의 이름이나 또는 "_top", "_blank", "_self" 등을 줄수 있다.

"_top": 해당하는 전체화면이나 표를 100%크기로 보여준다.

"_blank": 새 창을 열고 련결시켜준다.

"_self": 해당하는 창(왼쪽에 있었다면 왼쪽으로, 오른쪽에 있었다면 오른쪽으로, 전체화면이였다면 전체화면으로)으로 직접 련결시켜준다.

<a>끄리표를 리용하여 음악도 봉사받을수 있다. 음악이 있는 주소를 넣고 찰칵하게 되면 음악을 들을수 있다.

그 다음으로 많이 쓰이는것이 <form> + <input> 이다.

형식은 다음과 같다.

```
<form action="주소"><input type="형식"></form>
```

선택항목으로서는 <a>와 마찬가지로 target가 있다.

그 이외에도 method라는 전송형선택항목이 있다. 여기에는 post, get 등이 있다.

1.2.9. 복합칸끄리표(select)

<select>끄리표는 복합칸을 만드는 끄리표이다.

기본형식은 다음과 같다.

```
<select><option>현시내용</select>
```

만약 선택하려고 하는 항목이 많으면 <option>을 계속 추가해주면 된다.

즉

```
<select>
        <option>현시내용
        <option>현시내용
        <option>현시내용
</select>
```

<select>끄리표에 의하여 만드는 복합칸의 너비는 한행의 글자수에 따라 자동적으로 달라진다. 반면에 <select>와 비슷한 기능을 수행하는 <input>의 text형이나 <textarea> <iframe>등의 끄리표는 따로 너비를 조절하는 선택항목이 있다.

<select>꼬리표는 너비를 조절하는 선택항목은 따로 없지만 복합칸의 높이를 조절하는 선택항목(size)이 있으며 이외에도 name, value, multiple 등의 선택항목이 있다. name선택항목은 꼬리표의 이름을 나타내는 역할을 한다. multiple선택항목을 주면 다중선택이 가능하다.

1.2.10. textarea

textarea는 문자그대로 본문칸을 만드는 꼬리표이다.

형식은 다음과 같다.

```
<textarea>현시내용</textarea>
```

textarea와 비슷한 형식을 가지는것이 iframe꼬리표이다.

textarea는 모든 꼬리표가 무효화되는 반면에 iframe은 창안의 꼬리표가 적용된다.

textarea에서는 너비를 조절하는 선택항목은 width가 아니라 cols이다. 즉 cols="수자"라고 입력하면 된다. 여기서 수자의 단위는 1byte이다.

실 례:

```
<textarea cols="40">현시내용</textarea>
```

높이를 지정하는 선택항목은 rows이다.

실 례:

```
<textarea rows="7">현시내용</textarea>
```

textarea에 쓰이는 기본선택항목은 rows, cols, name들이다.

1.2.11. iframe

형식은 다음과 같다.

```
<iframe src="주소"></iframe>
```

이 꼬리표는 textarea와 외관상 매우 비슷하지만 그 특성은 서로 다르다. iframe은 다른 창을 펼치지 않고 그 창안에 《주소》에서 지적한 파일의 내용을 현시한다.

크기선택항목이 있는데 textarea처럼 rows, cols가 아니라 height, width이다. rows, cols가 절대단위인 반면에 width, height는 절대크기, 상대크기가 모두 가능하다.

만약 width, height선택항목을 주지 않는다면 width=300, height=150이 기정값으로 지정된다. 상대적크기도 가능하다.

또한 width=50%, height=40%와 같이 퍼센트로 표현할수도 있다.

일반적으로 상대값를 주면 해당하는 전체 화면이나 해당하는 표안에서의 퍼센트를 나타내는것이다.

width, height의 퍼센트선택항목은 iframe뿐만아니라 이 선택항목이 쓰이는 대부분의 꼬리표에서 사용가능하다. 즉 <image>, <table>, <tr>, <td>, <style>, <p>, <span>, <div> 등에서 사용할수 있다.

만일 현시하려는 다른 창의 크기가 iframe보다 크면 흘림띠가 생기게 된다. 흘림띠의 유무도 선택항목으로서 직접 지정해줄수 있다. 즉

"scrolling=no/yes/auto"

와 같은 형식으로 지정해준다. 이 선택항목을 주지 않는다면 기정값인 auto로 설정된다.

iframe에서 경계선이 보이는데 이것은 없앨수도 있다. 그것은 frameborder라는 선택항목을 리용하여 실현하는데 선택항목값은 yes이거나 no이다. 이 선택항목을 생략하면 기정값인 yes로 설정된다.

제3절. 창문분할과 표생성

1.3.1. 창문분할꼬리표(frame)

<frame>꼬리표는 창문을 나누는 꼬리표이다. 홈페지를 만들 때 한개의 창문에 여러개의 페지를 동시에 표시하려는 경우가 많다. 이것을 해결할 목적에서 나온것이 <frame>꼬리표이다.

실 례:

```
<html>
<head>
<title>창문분할페지</title>
</head>
<frameset cols 또는 rows="수자 또는 %, 수자 또는 % 또는 *">
<frame src="첫번째 프레임에 놓일 파일의 주소">
<frame src="두번째 프레임에 놓일 파일의 주소">
</frameset>
</html>
```

frameset가 들어간 페지는 다른 페지와 달리 <body>꼬리표를 생략한다.

<frameset>꼬리표는 말그대로 프레임의 형태를 설정해주는 역할을 하는 꼬리표이다. 창문을 세로로 나누려면 cols선택항목을, 가로로 나누려면 rows선택항목을 리용한다.(textarea와 선택항목이 같다. 차이점은 frameset의 cols, rows단위가 px이라는것이다.) 그리고《수자,*》의 뜻은 왼쪽창을 주어진 수자의 크기만큼 설정하고 나머지는 오

른쪽부분으로 한다는 의미이다.

실례:

```
<html>
<head>
<title>창문분할페지</title>
</head>
<frameset rows="120, *">
<frame src="frame1.html" scrolling=no noresize>
<frame src="frame2.html" scrolling=auto noresize>
</frameset>
</html>
```

프레임의 개수는 보통 2~3개정도로 할수 있다.

창을 하나 더 만들때에는 <frame>꼬리표를 하나 더 만들어서 추가하면 된다.

실례:

```
<html>
<head>
<title>창문분할페지</title>
</head>
<frameset rows="120, 170, *">
<frame src="frame1.html" scrolling=no noresize>
<frame src="frame2.html" scrolling=auto noresize>
<frame src="frame3.html" scrolling=auto noresize>
</frameset>
</html>
```

cols ,rows선택항목에 textarea와는 달리 상대적비률값을 줄수 있다. 그러나 <frame>꼬리표에서 상대비률값은 잘 쓰지 않는다. 그것은 해상도에 따라 다르게 보일수 있기때문이다. 그래서 대부분 경우 절대값크기(px)로 준다.

프레임들사이의 경계선의 존재여부를 결정하는 frameborder와 경계선두께를 설정해주는 framespacing선택항목이 있다. 그러므로 경계선을 없애려면 "frame border =no", "framespacing=0"으로 해주면 된다.

실례:

```
<html>
<head>
<title>창문분할페지</title>
```

```
</head>
<frameset cols="180, *" frameborder=no framespacing=0>
<frame src="frame1.html" scrolling=no noresize>
<frame src="frame2.html" scrolling=auto noresize>
</frameset>
</html>
```

<frame>에서 반드시 필요한것은 <frame src=주소> 부분이다. 여기서 주요선택항목인 name은 frame을 만들 때 name을 주지 않고 련결시키면 련결시켜주는 바로 그 창에서 이동하게 된다.(일반적으로 name을 생략하면 "target=_self"로 인식한다.)

target="해당하는 문서의 이름"을 주면 해당되는 문서가 있었던 위치에 련결시키는 것이다. 비록 련결시키는 문서는 왼쪽이지만 target을 해당하는 문서의 이름으로 취해주면 거기로 이동한다.

실례:

```
<html>
<head>
<title>프레임을 나누는 페지</title>
</head>
<frameset cols="180, *">
<frame name="a" src="frame1.html" scrolling=no noresize>
<frame name="b" src="frame2.html" scrolling=auto noresize>
</frameset>
</html>
```

이렇게 "frame1"에는 "A"라는 이름을 주고 "frame2"에는 "B"라는 이름을 주었다고 가정하면 련결페지에서는 즉 "frame1"페지에서는 target을 이동시키려고 하는 문서이름으로 주면 된다.

"frame1"은 왼쪽에 있고 "frame2"는 오른쪽에 있으므로 "frame1"의 target설정을 "B"로 하면 오른쪽에서 이동하게 된다.

noresize는 frame의 크기조절의 유무를 지정해주는 선택항목이다. 이 선택항목을 지정해주지 않으면 사용자들이 frame의 크기를 마음대로 바꿀수 있다.

scrolling선택항목은 <iframe>꼬리표에서도 언급했지만 흘림띠의 유무를 지정해주는 선택항목이다. "scrolling=yes/no/auto"중에서 선택을 하면 된다. 기정값은 auto로 되여있다.

여백을 주기 위해서는 "marginwidth=수자", "marginheight=수자" 형식으로 지정해주면 된다. 첫번째형식은 가로여백을 정하는것이고 두번째형식은 세로여백을 지정하는

것이다. 지정하지 않으면 알아서 적당히 여백을 정해준다. <table>꼬리표의 cellpadding선택항목과 비슷하다.

실 례:

```
<html>
<head>
<title>프레임을 나누는 페지</title>
</head>
<frameset cols="180, *">
<frame src="frame1.html" scrolling=no noresize>
<frameset rows="50%,50%">
<frame src="frame2.html" >
<frame src="frame3.html" >
</frameset>
</frameset>
</html>
```

1.3.2. 령역꼬리표(area)

홈페지사용에서 보게 되면 하나의 그림에 여러개의 련결이 되여있는것을 볼 때가 있다. 일반적으로 그림의 련결은 그림의 어느 곳을 찰칵하든 하나의 파일만이 현시되도록 되여있다. 그러나 <area>꼬리표를 리용하면 그림의 어떤 부분을 찰칵하는가에 따라 서로 다른 파일이 현시되도록 할수 있다.

area련결은 두개의 꼬리표를 써서 나타낸다. 즉 형식은 다음과 같다.

```
<image src=그림주소 usemap="#지도의 이름">
…
<map name="지도의 이름">내용
<area shape=모양값 coords=자리표 href=련결시킬 주소>
</map>
```

그림꼬리표에 대해서는 우에서 취급하였는데 그 꼬리표에 usemap라는 선택항목이 추가되여있다. 《usemap》라는 선택항목은 area련결을 알리는 명령으로서 《지금부터 이 그림을 지도(map)로 삼겠다.》라는 뜻이다.

<map>는 그림을 련결령역으로 지정하기 위한 시작꼬리표이다. 이 꼬리표에는 name선택항목이 있다. name선택항목값은 우의 image꼬리표에서 지정해주었던 이름을 넣어주어야 한다.

<area>는 그림의 어떤 위치에 련결하겠는가 하는 area련결과 관련한 구체적인 설정을 해주는 부분이다. 이것은 <map>와 </map>사이에 들어간다. 이 꼬리표의 선택항목에는 shape와 coords외에 <a>꼬리표와 똑같은 선택항목인 href와 target이 있다.

사용자는 자기가 찾아보려는 부분에서 마우스지시자가 손가락모양으로 변할 때 찰칵하여 요구하는 자료내용을 볼수 있다.

실례:

```
<image src=starcraft.gif usemap="#sample" border=0>
<map name="sample">
<area shape=rect coords="56,83,112,129" href="mis.html"
target="_blank">
<area shape=rect coords="17,93,52,126" href="tank.html"
target="_blank">
<area shape=rect coords="154,70,184,97" href="scv.html"
target="_blank">
</map>
```

① shape선택항목

한 그림에서 련결시켜주려는 모양을 정해주는 선택항목이다. 이 선택항목에는 rect, circle, poly가 있다.

rect는 경계선이 직사각형으로 나오고 circle은 원형, poly는 사용자가 요구하는 임의의 다각형으로 나온다.

② coords선택항목

coords선택항목은 련결령역을 정해주는 역할을 한다. 즉 《어디서부터 어디까지 마우스를 찰칵하면 어느 파일로 넘어간다》라는 기능을 가지는 명령이다. 그 범위는 자리표값으로 나타내는데 단위는 px이다.

자리표점의 기본형식은 (x축값, y축값)이다. 해당 그림의 왼쪽우의 꼭두점이 기준점자리표 (0,0)으로 설정되여있다.

직사각형일 때에는 우선 련결시킬 부분의 왼쪽우에 꼭두점을 적어준 다음 오른쪽아래의 꼭두점을 찍어주면 된다. 만약 왼쪽우의 자리표가 (150,30)이고 오른쪽아래의 자리표가 (200,50)이라면 cords= "150, 30, 200, 50"으로 주면 된다.

원형일 때에는 원의 중심자리표를 먼저 적어주고 반경값을 넣어주면 된다. 만약 중심점이 (100,40)이고 반경이 30이면 coords="100, 40, 30"으로 주면 된다.

다각형일 때에는 시계방향으로 원하는 자리표값을 넣어주면 된다. 자리표점이 5개라면 5각형이 되고 8개라면 8각형이 된다. 만약 시계방향으로 (100,5), (120,10), (130,15),(125,20),(110,10)를 주었다면 cords= "100,5, 120, 10, 130, 15, 125, 20,

110,10"으로 주면 된다.

1.3.3. 검색기능꼬리표(meta)

사용자가 입력한 검색어를 검색엔진이 찾을 때 제일 먼저 참고하는것이 바로 이 부분이다. 즉 검색어를 제공하는 기능을 가지는 꼬리표이다.

기본형식은 다음과 같다.

```
<meta name=이름값 content=내용>
```

<meta>꼬리표는 <head>와 </head>사이에 들어가며 열람기에 표현되지는 않는다.

실례:

```
<html>
<head>
<title>HTML 홈페지</title>
<meta name="Subject" content="tag & java">
<meta name="Description" content="HTML의 모든것, HTML의
집대성!" >
<meta name="Keywords" content="HTML강의,HTML강좌,강좌, 강의,
홈페지만들기, HTML, JavaScript, CSS, JAVA Applet, DHTML">
</head>
<body>
…
</body>
</html>
```

- <meta name="Subject" content="tag & java">

해당하는 페지에 관한 제목을 설정하는 부분인데 어떠한 주제를 다루는가를 입력해 놓은 부분이다. 검색로보트로 찾기가 훨씬 쉽다. 여기서는 tag & java라고 넣었다.

name의 속성에는 keywords, subject, description 등이 있다. 그외에도 title, author,date 등이 더 있지만 우에 있는것들만 알아도 충분하다.

우에 name값은 subject이므로 content에는 거기에 맞는 내용을 써줘야 한다.

- <meta name="Description" content="HTML의 모든것, HTML의 집대성!" >

이것은 검색엔진이 해당하는 페지를 찾은 다음 간단한 요약을 표시해 줄 때 나타나는 내용이다. 따라서 자신의 홈페지에 대해 가장 알맞고 효과적인 설명을 간명하게 만들어 주어야 한다.

- <meta name="Keywords" content="HTML강의, HTML 강좌, 강좌, 강의, 홈페지만들기, HTML, JavaScript, CSS, JAVA애플레트, DHTML">

이것은 자신의 홈페지를 찾을 때 해당하는 검색어를 지정해주는 내용이다. name의 keyword선택항목이 바로 그러한 검색어를 지정해주는 기능을 한다. content란에는 해당하는 검색어를 넣어야 한다.

- <meta http-equiv=값 content=내용>

http-equiv선택항목의 값에는 주로 'Content-Type'과 'Refresh'가 쓰인다.

http-equiv="Content-Type" 이라는 값을 넣으면 이것은 서체수정기능을 수행한다.

실 례:

<meta http-equive="Content-Type"
content="text/html;charset=big5">

페지에 이러한 명령을 넣으면 본문서체를 자동적으로 big5라는 서체로 수정해주는 기능을 한다.

refresh란 말그대로 새롭게 하는 기능을 하는것이다
그렇다면 http-equiv="Refresh"라고 넣었을 때에는 구체적인 내용을 content란에 써주어야 한다.

실례:

<meta http-equive="Refresh" content="수자;url=새로운 주소">

content의 수자는 새로운 페지를 적재할 때 걸리는 시간으로서 sec (초)단위를 가진다. 그리고 url은 새로 이동하게 될 주소를 넣는곳이다.
meta의 refresh값은 JavaScript의 setTimeout()함수와 동일한 기능을 가진다.
refresh값을 페지마다 주면 자동현시기능을 가지게 된다. 즉 일정한 시간이 지나면 자동적으로 계속 다른 주소의 페지를 보여준다.

1.3.4. 표생성꼬리표(table)

이 꼬리표는 여러개의 행과 렬로 된 표를 작성한다.
형식은 다음과 같다.

```
<table border=2> </table>
```

<table>이란 말그대로 표의 시작을 알리는 꼬리표이다. 따라서 표가 끝나는 위치에서 반드시 </table>을 해주어야 한다.
이 꼬리표에는 다음과 같은 선택항목들이 있다.

- border

  표 경계선의 굵기를 지정한다.

- width

너비를 지정한다. 절대크기, 상대크기를 다 나타낼수 있다.

- height

  높이를 지정한다. 절대크기, 상대크기를 다 나타낼수 있다.
- cellspace

  세포의 테두리와 본문사이의 간격을 지정한다.
- cellpadding

  세포들사이의 간격을 지정한다.
- bgcolor

  세포의 배경색을 지정한다.
- background

  표의 배경그림의 위치를 지정한다.
- bordercolor

  표의 경계선의 색을 지정한다.

<tr>꼬리표는 Table Row의 줄임말로서 말그대로 표의 한 행을 정의하는 꼬리표이다. 그러므로 2개행으로 된 표를 작성하자면 <tr>꼬리표를 두개 사용해야 한다.

<td>꼬리표는 Table Detail의 줄임말로써 하나의 세포를 만드는 꼬리표이다.

한 행에 3개의 칸이 있는 표이라면 <tr>를 한번 쓰고 <td>는 3번 쓴다.

<table>과 <tr>, <td>의 배치순서를 보자.

우선 표를 먼저 만들어야 하므로 <table>를 쓰고 그다음 한개의 행을 만들어야 하므로 <tr>를 입력한 다음 마지막으로 그 행에다가 한개 이상의 칸을 만들어야 하므로 <td>를 입력한다. 따라서 실행순서는 <table> -> <tr> -> <td>이다. <td>는 <tr>의 령역에 포함되여있다. 그리고 <tr>은 <table>의 령역에 또 포함된다.

따라서 형식은 다음과 같다.

```
<table><tr><td>내용</td></tr></table>
```

세포에 관한 꼬리표들의 선택항목에는 다음과 같은것이 있다.

- align

  align선택항목은 세포안의 본문을 왼쪽, 오른쪽, 중심에 배치한다.

  align=left는 왼쪽에, center는 중심에, right는 오른쪽에 놓이게 한다.
- valign

  valign선택항목은 세포안의 본문을 웃단, 중심, 아래단에 배치한다.

  valign=top는 웃단에, middle은 중간에, bottom은 아래단에 놓이게 한다.
- bgcolor

  세포안의 배경색을 지정한다.

- rowspan

  렬수를 지정한다.
- colspan

  행수를 지정한다.
- nowrap

  세포안에서 자동적인 행바꾸기를 금지한다.

그리고 <td>안의 내용에는 글자뿐아니라 다른 꼬리표를 넣어서 여러가지 기능을 수행할수 있다.

제 2 장. STYLE SHEET 언어

제1절. CSS의 기초

2.1.1. 개념

CSS란 Cascading Style Sheet의 줄임말로서 웨브페지를 만들 때 꼬리표만으로는 자신이 표현하려고 하는것을 나타내기 어려운 경우가 있을 때 그것을 실제로 가능하게 해주는 언어이다.

CSS의 우점을 보자

① 문서의 통일성

례를 들어 표에 들어가는 서체를 모두 크기는 《2》, 색갈은 푸른색, 서체종류는 《청봉체》로 만들려고 한다. 그렇다면 표의 매 칸마다 <font size=2 color=blue face=청봉>라고 써줘야 한다. 표가 한 두개라면 모르겠지만 실제로 웨브페지를 설계할 때 많은 표를 작성해야 하므로 이 서체꼬리표들을 일일히 넣어주는데 걸리는 시간랑비, 로력랑비가 많아진다. 그렇다면 모든 표에 들어가는 서체를 《한번의 명령》으로 될수록 통일시켜주어야 할 필요성이 제기된다. 이것은 CSS를 통해 실현할수 있다.

실례:

-꼬리표만 리용한 경우

```
<html>
<head>
<title>꼬리표사용</title>
</head>
<body>
<table border=1><tr><td>
<font color=blue size=2 face=청봉>세포1-1</font>
</td>
<td>
<font color=blue size=2 face=청봉>세포1-2</font>
</td></tr>
<tr><td>
<font color=blue size=2 face=청봉>세포2-1</font>
</td>
<td>
```

```
<font color=blue size=2 face=청봉>세포2-2</font>
</td></tr></table>
</body>
</html>
```

- CSS를 리용한 경우

```
<html>
<head>
<title>CSS사용</title>
<style type="text/css">
td {font-size:12px;font-family:청봉;color:blue}
</style>
</head>
<body>
<table border=1><tr>
<td>세포1-1</td><td>세포1-2</td>
</tr>
<tr>
<td>세포2-1</td><td>세포2-2</td>
</tr></table>
</body>
</html>
```

실행화면은 똑 같은데 서체에 관한 설정에서 일반꼬리표로 4번 표시하였으나 CSS는 단 한번의 명령으로 같은 효과를 내게 한다.

② 문서의 다양성

이전의 HTML에서 불가능한 여러가지 표현을 가능하게 해준다. 다시 말하여 그림을 원본보다 희미하게 해준다든가, 글자에 그림자를 생기게 한다든가, 한 행간격의 사이를 조절할수 있는 다양한 표현이 가능하다.

③ 적재시간단축

입력되는 문자가 HTML보다 적으므로 그만큼 적재시간도 빨라진다.

꼬리표에 대해서만 알고있으면 CSS를 습득하는것은 어렵지 않다. 그것은 CSS는 꼬리표에 부가적인 속성을 더해주는것 뿐이기때문이다. 따라서 문법자체를 따로 공부해야 하는 JAVA스크립트보다는 훨씬 배우기 쉽다. 이러한 우월성을 가지고있는 CSS는 현재 CSS2까지 나왔다.

2.1.2. STYLE SHEET 의 표기방식

CSS에 대하여 실례를 들어 설명해보자.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
<!--
font {color:red ;font-size:20px ;background:yellow}
--> </style>
</head>
<body>
<font>나는 할수 있다.</font>
</body>
</html>
```

① CSS의 기본형식

CSS의 기본형식은 다음과 같다.

선택항목1:선택항목1의 값; 선택항목2:선택항목 2의값....

CSS는 하나의 선택항목안에 또다른 선택항목값을 넣을수도 있다.

실례:

```
color:blue;height:10;filter:shadow(color:red)
```

선택항목에 또다른 선택항목을 넣을 때에는 괄호 ( )를 써서 표시해준다. 물론 아무 선택항목이나 괄호를 치는것이 아니다. 그러한 속성을 지원하는 선택항목에만 한다.

② CSS의 표현방식

크게 2가지 방법이 있다. 하나는 매몰(embedded)방식이고 다른 하나는 꼬리표안에 종속되여서 쓰이는 직접(inline)방식이다.

매몰방식은 우의 례제와 같이 CSS부분을 따로 지정해주는것이다.

형식은 다음과 같다.

```
  <style type="text/css">
  <!-
선택항목1:선택항목1값;선택항목2:선택항목2값...}
  -->
  </style>
```

이 방식은 대체로 CSS부분이 <head>와 </head> 사이에 놓이며 문서전체에 영향을 미친다. 한번만 설정해주면 문서가 끝날 때까지 효력을 낸다. CSS의 우점에서 설명한바와 같이 이 방식은 문서의 통일성을 실현한다.

실례:

```
<html>
<head>
<title>CSS례제</title>
<style type="text/css">
<!--
font {color:red ;font-size:20px ;background:yellow}
-->
</style>
</head>
<body>
<font>나는 할수 있다</font><br>
<font>나를 믿는다</font>
</body>
</html>
```

웃줄의 《나는 할수 있다》와 아래줄의 《나를 믿는다》는 각각 <font>꼬리표로 씌여졌지만 모두 같은 서체격식이 적용된다. 그것은 CSS에서 모든 서체에 대한 조정을 하고있기때문이다

직접방식은 CSS부분을 특정꼬리표안에 넣고 리용하는 방식이다.

형식은 다음과 같다.

```
<적용될 꼬리표 style="선택항목1:선택항목1값;선택항목2:선택항목2값...">
```

이 방식은 한 꼬리표에 종속되므로 해당 꼬리표의 령역안에서만 CSS부분이 적용된다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
</head>
<body>
```

```
<font style="color:red ;font-size:20px ;background:yellow">나는 할수  있다</font><br>
<font>나를 믿는다</font>
</body>
</html>
```

《나는 할수 있다》에만 서체격식이 적용되고 아래의 행에는 아무런 변화도 없다. 실례를 통하여 알수 있듯이 매몰방식과 달리 해당 꼬리표에만 적용된다.

제2절. 서체와 배경 조정

2.2.1. 서체조정

CSS의 서체관련선택항목에는 font-size, font-family, color, font-weight, font-style, text-transform, text-decoration 등이 있다.

```
<style type='text/css'>
적용될 꼬리표     {font-size:값;font-family:값;font-weight:
값;font-style:값;text-transform:   값;text-decoration:값}
</style>
```

- 서체크기(font-size)

기본단위값은 px이다. <font>꼬리표보다 훨씬 더 세밀하게 크기를 조정할수 있다. <font>는 size=1~7까지 밖에 조절할수 없지만 CSS의 font-size선택항목값은 무한대이다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
font {font-size:30}
</style>
</head>
<body>
<font>나는 할수 있다</font>
```

```
</body>
</html>
```

수자만 써주면 단위는 px로 인식한다. px외에도 pt라는 단위가 있는데 이 단위를 쓸 때에는 뒤에 단위를 붙여주어야 한다. 실례로 《font-size:20pt》로 하면 된다. pt는 px보다 큰 단위이다.

- 서체종류(font-family)

서체종류를 선택하는 선택항목이다.

그 형식은 다음과 같다.

font-family: 서체이름

CSS의 font-family선택항목이 <font>의 face와 다른점은 코드에 여러개의 서체를 써줄수 있다는 점이다.

어떤 서체를 써줄 때 그 서체가 해당 사용자의 콤퓨터에 없으면 콤퓨터는 무조건 고직체로 처리한다. 하지만 font-family값을 줄 때 어떤 서체가 콤퓨터에 없을 경우 대신 다른 서체를 예비로 지정해줄수 있다. 실례로 《font-family:Comic Sans Ms,Arial》로 준다면 기본서체가 Comic Sans Ms이지만 없을 경우에는 Arial서체가 적용되게 된다는것이다.

- 서체의 굵기(font-weight)

<b>꼬리표의 역할과 같다.

font-weight선택항목은 bold와 thin의 선택항목값을 가진다.

- 서체의 형식(font-style)

서체의 형식을 지정해줄수 있는데 그 값은 normal과 italic이다.

normal은 보통서체를 나타내고 italic은 경사체를 나타내는것으로서 <i>꼬리표와 류사하다. 그러므로 font-style:normal 또는 font-style:italic으로 써준다.

- 대소문자변환(text-transform)

대소문자변환선택항목이므로 영문에만 적용된다. 그 값에는 아래와 같은 값들이 있다.

uppercase: 모두 대문자로 변환

lowercase: 모두 소문자로 변환

capitalize: 띄여쓰기할 때 첫 글자를 대문자로 변환

- 줄긋기(text-decoration)

본문에 줄을 긋는 기능이다. 선택항목값에는 다음과 같은것들이 있다.

underline: 문자에 밑줄긋기
overline: 웃줄긋기
line-through: 가운데 줄긋기

- 색조절(color)

이것은 적용될 꼬리표의 글자색을 조절하는 선택항목이다. 그 값은 blue, red, black같은 영문이름도 될수 있고 0000ff, ff0000, 000000과 같이 표기할수도 있다. 즉 《color:색갈이름》과 같은 형식이다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
font {color:red}
</style>
</head>
<body>
<font>나는 할수 있다</font>
</body>
</html>
```

HTML에서는 단추의 글자색갈, textarea의 글자, select꼬리표의 글자를 바꿀수 없지만 여기서는 이 모든것이 가능하다.

2.2.2. 배경조정

CSS에서 배경에 관한 선택항목은 background-color, background-image, background-repeat 그리고 background-attachment 등이 있다. 그중 background-color는 다른 선택항목들과 같이 쓸수 없다는것을 고려하면 그 형식은 다음과 같다.

```
<style type='text/css'>
 적용될 꼬리표 {backgound-color:값(또는 backgound-image:값;backgound-repeat:값;background-attachment:값)}
</style>
…
 <적용될 꼬리표>
```

역시 서체조정과 마찬가지로 선택항목전부를 쓸 필요없이 원하는 선택항목만 써주면 된다.

- 단색배경(background-color)

단색배경을 줄 때 쓰이는 선택항목이다. 그 형식은 《background-color:색갈》이다.

실례:

```
<html>
```

```
<head>
<title>CSS 례제</title>
<style type="text/css">
font {background-color:yellow}
</style>
</head>
<body>
<font>나는 할수 있다</font>
</body>
</html>
```

background-color선택항목을 <body>꼬리표에 주면 화면전체에 색갈이 적용된다.

- 그림배경

그림배경에 쓰이는 선택항목들은 많다. 그중에서 가장 기본적인것은 background-image이다.

형식은 다음과 같다.

```
background-image:url(그림주소)
또는
background:url(그림주소)
```

이 선택항목은 <body>꼬리표에도 적용할수 있다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
body {background-image:url(back.gif)}
</style>
</head>
<body>
나는 할수 있다
</body>
</html>
```

그림이 배경범위보다 크기가 작은 경우 반복되여 나오는데 그것을 조절할수 있는 선택항목이 background-repeat이다. 이 선택항목은 background-image선택항목과

함께 쓰인다.

이것의 선택항목값에는 다음과 같은것들이 있다.

repeat-x : x축으로 반복이 된다.

repeat-y : y축으로 반복이 된다.

norepeat : 배경그림이 반복되지 않는다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
body
{background-image:url(back.gif);background-repeat:repeat-
x}
</style>
</head>
<body>
나는 할수 있다
</body>
</html>
```

background-attachment선택항목도 그림배경에 관한 선택항목이다. 문자그대로 배경을 붙이는 역할을 하는데 화면이 커서 흘리게 하여 열람하는 경우 배경은 따라 움직이지 않고 고정시킨다. 선택항목값은 fixed이다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
body {background-image:url(back.gif);background-
attachment:fixed}
</style>
</head>
<body>
내나라 제일로 좋아</body>
</html>
```

제3절. 위치, 단락, 유표 조정

2.3.1. 위치조정

위치조정이란 어떤 내용을 자신이 원하는곳에다가 마음대로 놓을수 있는것을 말한다. CSS의 위치에 관한 선택항목에는 positon, z-index, left 그리고 top 등이 있다.

그 형식은 다음과 같다.

```
<style type='text/css'>
 적용될 끄리표 {position:값;z-index:값;top:값;left:값}
</style>
```

- 위치지정선택항목(positon)

이것은 위치를 지정하는 선택항목이다. 선택항목값에는 absolute와 relative가 있다. 그러므로 《position:absolute》로 하면 위치를 절대적으로 지정하는것이고 《position:relative》는 상대적으로 위치를 지정해주는것이다.

절대적인 위치지정을 한 경우 정확한 위치설정은 top과 left선택항목에서 한다. 그러므로 position선택항목은 top과 left선택항목이 항상 같이 쓰인다. 즉 《opsition;absolute;top:값;left:값》으로 된다. 위치단위는 px이다.

절대위치의 자리표는 화면전체에 해당한다. 여기서는 화면의 왼쪽웃구석이 기준점으로 된다.

례를 들어 아래로 10px만큼만 내리고싶다면 top:10으로 주면 된다. 또 거기에다 왼쪽으로부터 오른쪽으로 30px만큼만 이동하고싶으면 left:30으로 주면 된다. 즉 《position:absolue;top10;left:30》와 같이 하면 된다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
image {position:absolute;top:150;left:100}
</style>
</head>
<body>
<image src=kitty.gif>
</body>
```

```
</html>
```

상대적인 위치지정은 화면전체가 아니라 어떤 특정한 다른것을 기준점으로 삼아서 거기에 상대적으로 위치를 지정하는 값이다. 형식은 《position:relative》와 같다. 물론 상대적인 위치지정도 top와 left선택항목이 같이 쓰인다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
font {position:relative;top:10;left:15}
</style>
</head>
<body>
HTML 작성
<font>나는 할수 있다</font>
</body>
</html>
```

《HTML 작성》이라는 본문을 기준으로 아래로 10px, 오른쪽으로 15px만큼 되는 위치에 《나는 할수 있다》라는 본문이 나타난다.

절대위치지정에서는 화면의 왼쪽웃구석이 기준점으로 되므로 더이상 우로, 왼쪽으로 갈수 없지만 상대위치는 기준점이 임의의 위치에 있으므로 기준점보다 웃쪽으로 또는 왼쪽으로 갈수 있다. 이때에는 top와 left값을 부의 부호(-)로 준다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
font {position:relative;top:-10;left:-15}
</style>
</head>
<body>
HTML 작성
<font>나는 할수 있다</font>
</body>
```

```
</html>
```

- 순서결정(z-index)

상대위치나 절대위치로 지정하게 되면 내용끼리 겹쳐지게 되는 경우가 있다. 즉 그림과 글자가 겹쳐지던가 아니면 표와 그림이 겹쳐질수 있다. 이러한 경우 z-index선택항목을 리용하여 그 층의 순서를 결정할수 있다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
a {position:absolute;top:15;left:15; z-index:2}
font {position:absolute;top:15;left:15; z-index:1}
</style>
</head>
<body>
<a href=http://kr.kp.naenara.com target=_blank>'내나라' </a>
<font>나는 할수 있다</font>
</body>
</html>
```

우의 례제는 <a>끄리표와 <font>끄리표를 겹쳐 놓았을 때 <a>끄리표와 <font>끄리표의 우선권을 결정하는 례제이다. <a>끄리표는 z-index가 2이고 <font>는 z-index가 1이므로 <font>끄리표가 적용된것이 먼저 놓이고 <a>끄리표가 적용된것이 우에 놓이게 된다. 그러므로 글자(내나라)우로 마우스를 가져갔을 때 찰칵이 가능하다.

실례:

우의 실례에서 순서를 바꿔보자.

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
a {position:absolute;top:15;left:15; z-index:1}
font {position:absolute;top:15;left:15; z-index:2}
</style>
</head>
<body>
```

```
<a href=http://kr.univ.com target=_blank>대학</a>
<font>나는 할수 있다</font>
</body>
</html>
```

이번에는 <a>가 먼저 놓이고 <font>가 마지막에 놓였으므로 <font>의 기능이 우선권을 가지고 <a>는 밑에 배치된다. 따라서 마우스를 가져가도 찰칵이 안된다.

2.3.2. 단락조정

단락의 구체적인 속성을 조절해주는 기능이다. word로 문서를 작성할 때 글자사이의 간격을 조절, 행간격의 조절, 배치를 오른쪽으로 또는 중심으로 할것인가를 하는것을 선택하는데 CSS를 리용할수 있다.

- 행간격조절(line-height)

line-height는 웃줄의 글자와 아래줄의 글자사이의 간격을 값으로 조절하는것이다. 값은 수값으로도 나타낼수 있고 퍼센트로 나타낼수 있으며 px 등의 단위로도 나타낼수 있다. 즉 line-height:수자, line- height:수자%, line- height: 수자px의 세가지 방법이 있다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
font {line-height:3}
</style>
</head>
<body>
<font> '내나라 제일로 좋아'</font>
</body>
</html>
```

우의 실례에서 'line-height:3'은 글자크기의 3배만큼 띄우라는 의미이다. 그러나 웃줄의 글자웃부분에서 시작해서 3배이다.

보다싶이 첫줄의 글자웃부분부터 해당글자의 배수를 재서 나타내게 된다. 이것은 HTML의 <p>나
보다도 훨씬 구체적으로 조정할수 있다는것을 알수 있다. 여기서 수자는 자연수뿐아니라 소수도 가능하다.

- 글자간격조절(letter-spacing)

글자와 글자사이의 간격을 조절할수 있는 선택항목이다. 이 값은 우의 line- height

와는 달리 단위적 표현만이 가능하다.

실례:

```
<html>
<head>
<title>CSS 레제</title>
<style type="text/css">
font {letter-spacing:5px}
</style>
</head>
<body>
<font> 내나라 제일로 좋아 </font>
</body>
</html>
```

여기서 0px로 준다면 글자사이의 간격은 없어지게 된다.

- 본문배치(text-align)

HTML에서 align에 해당하는 선택항목으로서 어떤 내용을 어느 방향으로 배치시킬것인가를 정하는것이다. 선택항목값에는 left, center, right가 있다. 그런데 이 선택항목은 <font>끄리표에는 적용되지 않는다. 정렬시키는 선택항목이므로 <p>나 <div>같은것에 적용된다.

- 문서의 첫글자들여쓰기 조절(text-indent)

문서를 작성할 때 단락의 첫글자는 약간 들여쓰는데 얼마만큼 들여쓰겠는가를 설정하는 선택항목이다. 이것도 역시 <font>에서는 쓰이지 않다.

Px 등의 단위로 나타내는 방법과 퍼센트로 나타내는 방법이 있다. text-indent:10px 라면 10px만큼 들여쓰는것이고 text-indent:10%라면 전체화면의 10%만큼 들여쓰는것이다.

- 문서의 여백조절(margin-left, margin-right, margin-top, argin-bottom)

공간을 얼마만큼 남길것인가를 정해주는 선택항목이다.

margin의 종류는 다음과 같다.

```
margin-left:값 //왼쪽의 여백을 조절한다
margin-right:값 //오른쪽의 여백을 조절한다
margin-top:값 //웃쪽의 여백을 조절한다
margin-bottom:값 //아래쪽의 여백을 조절한다.
```

값에는 px 또는 퍼센트를 단위로 줄수 있다. 따라서 10px라면 10px만큼 여백을 준다는것이고 10%라면 화면크기의 10%만큼 여백을 준다는것이다. 따라서 퍼센트로 주면

같은 값이라도 화면의 크기에 따라 여백이 달라질수 있다.

2.3.3. 유표조정

마우스를 특정한 내용우에 올려다놓았을 때의 마우스유표모양을 임의로 지정해줄수 있는 선택항목이다. 마우스를 글자우에 놓았을 때의 마우스모양은 《I》의 형태로 되고 하이퍼련결우에서는 손가락모양으로 된다. 그 외의 령역에서는 화살표모양으로 된다.

유표를 조절하는 선택항목은 cursor이다.

- 우-왼쪽 화살표(nw-resize)
  마우스유표를 우-왼쪽으로 향하는 화살표로 지정한다.
- 우-오른쪽 화살표(ne-resize)
  마우스유표를 우-오른쪽으로 향하는 화살표로 지정한다.
- 아래-왼쪽 화살표(sw-resize)
  마우스유표를 아래-왼쪽으로 향하는 화살표로 지정한다.
- 아래-오른쪽 화살표(se-resize)
  마우스유표를 아래-오른쪽으로 향하는 화살표로 지정한다.
- 물음표(help)
  마우스유표를 물음표로 지정한다.
- 이동표(move)
  마우스유표를 이동하는 표시모양으로 지정한다.
- 오른쪽 화살표(e-resize)
  마우스유표를 오른쪽으로 향하는 화살표로 지정한다.
- 웃쪽 화살표(n-resize)
  마우스유표를 웃쪽으로 향하는 화살표로 지정한다.
- 아래쪽 화살표(s-resize)
  마우스유표를 아래쪽으로 향하는 화살표로 지정한다.
- 왼쪽 화살표(w-resize)
  마우스유표를 왼쪽으로 향하는 화살표로 지정한다.
- 본문입력상태(text)
  마우스유표를 《I》모양으로 지정한다.
- 십자(crosshair)
  마우스유표를 십자가 형태로 지정한다.
- 모래시계(wait)
  마우스유표를 대기형태의 모래시계로 지정한다.
- 손모양(hand)
  마우스유표를 손모양으로 지정한다.

제4절. 범위, 경계선 조정

2.4.1. 범위조정

범위조정은 특정꼬리표의 영향을 받는 시각적인 범위를 조정하는 선택항목기능이다. 여기서 말하는 범위조정이라는 표현은 크기조정과 의미가 다르다.

실례로 <input>로 단추를 만들었는데 범위조정을 리용해서 단추를 크게 만들었다고 하자. 이 경우는 단추가 커졌으므로 범위와 크기의 의미는 같지만 <font>꼬리표의 글자를 범위조정으로 확장하면 글자가 커지는게 아니라 그 주변범위가 확장된다. 범위조정은 바로 이런 개념이다. 만약 글자의 범위를 조정한다면 글자의 크기는 그대로 있고 원래 글자가 차지하고있던 공간이 커진다는 뜻이다. 그리고 단추나 본문구역(textarea)같은 것은 차지하고있던 공간자체가 원래 그 꼬리표의 범위이므로 범위의 값을 크게 주면 크기 또한 커진다.

범위를 조정하는 선택항목에는 width와 height가 있다.

범위조정의 기본형식은 다음과 같다.

```
width:값;height:값
```

물론 width나 height중 하나만 써도 된다. 값의 단위는 기본적으로 px이고 상대크기를 표현하는 퍼센트(%)를 단위로 할수도 있다.

실례 1:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
font {width:150}
</style>
</head>
<body>
<font>나는 할수 있다</font>나를 믿는다
</body>
</html>
```

실례 2:

```
<html>
```

```
<head>
<title>CSS 례제</title>
<style type="text/css">
font {width:150;height:100}
</style>
</head>
<body>
<font>나는 할수 있다</font>나를 믿는다
</body>
</html>
```

실례 3:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
font {width:70%;height=50%}
</style>
</head>
<body>
<font>나는 할수 있다</font>나를 믿는다
</body>
</html>
```

width, height선택항목을 리용하는 모든 끄리표(<table>, <tr>, <td>, <iframe> 등) 들은 값을 퍼센트로 주면 상대범위이므로 화면의 크기에 따라 시각적으로 차이난다.

물론 해상도에 따라서도 차이가 난다. 그것은 800*600해상도일 때 50%로 주면 크기는 400px이지만 1024*768일 때는 512px가 되기때문이다.

따라서 홈페지를 만들 때 여러 해상도를 고려해서 적당한 상대비률을 지정해주는것도 중요한 문제이다. 물론 상대비률을 무시하고 px 등의 절대비률로만 리용해서 만든 홈페지도 있다. 해상도가 어떻든 화면크기가 어떻든 항상 똑같은 범위로 해주는것도 좋지만 해상도가 높은 화면에서 보는 경우 화면이 좀 작아보이는 단점이 있다. 반면에 100%로 상대크기를 지정해주면 해상도가 어떻든 전체화면을 보여줄수 있다. 다만 전체 화면이 현시되였지만 해상도에 따라 범위가 달라지므로 어떤 해상도에서는 화면이 이상하게 현시될수 있다는것이다. 따라서 모든 해상도를 고려해서

상대비률를 적절히 주도록 해야 한다.

범위조정의 선택항목은 단독으로 보다는 다른 선택항목과 함께 쓰이는 경우가 많다.

실례:

```
<font style="width:150;height:50;
            background-color:orange">나는 할수 있다</font>
<font style="width:150;height:50;
      background-color:orange;text-align:center">나는 할수 있다</font>
<font style="width:150;height:50;
      font-size:15;filter:shadow(color:red)">나는 할수 있다</font>
```

실제로 CSS는 <font>꼬리표보다는 <select>나 <input>, <a>, <textarea> 등에 더 많이 쓰인다.테두리선

실례:

```
<input style="width:100;height:40;background-
color:white;text-align:left" type=button value="멋있다">
```

우의 실례와 같이 <input>꼬리표를 비롯한 다른 꼬리표들은 서체와는 달리 범위자체가 그 꼬리표의 령역이므로 범위를 크게 해주면 크기가 커지게 된다.

2.4.2. 경계선조정

이것은 경계선을 임의의 형태로 만들수 있는 기능이다.

<table>꼬리표에서는 경계선의 속성을 조절하는 선택항목이 따로 있었지만 기타 꼬리표에는 그런 선택항목이 없었다. 바로 이것을 CSS가 가능하게 해준다. <table>의 경계선은 물론 <input>, <font>, <iframe>, <textarea> 등 모든 꼬리표에 대한 경계선을 CSS선택항목으로 조절할수 있다

- 경계선의 모양 지정 (border-style)

경계선의 모양을 어떻게 지정할것인가를 정하는 선택항목이다.

inset: 경계선의 안쪽부분이 안으로 움푹 들어간것처럼 만들어주는 선택항목이다.

outset: 경계선의 안쪽부분이 밖으로 나온것처럼 만들어주는 선택항목이다.

double : 경계선을 2중으로 만들어주는 선택항목이다.

solid: 직선으로 된 경계선을 만들어주는 선택항목이다.

dotted: 점선으로 된 경계선을 만들어주는 선택항목이다.(Internet Explorer 5.5 이상에서 실행)

dashed: 풀이표형식으로 된 경계선을 만들어주는 선택항목이다.(Explorer 5.5이상에서 실행)

ridge: 경계선을 액자형식으로 만들어주는 선택항목이다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
font {border-style:solid}
</style>
</head>
<body>
<font>나는 할수 있다</font>
</body>
</html>
```

dotted, dashed는 Explorer 5.5이상에서만 제대로 실행된다. 5.5이하에서는 dotted나 dashed를 solid형으로 인식한다.

- 경계선두께조절(border-width)

이 선택항목을 주지 않으면 기정값인 《border-width:4》로 인식한다.

inset나 outset로 설정된 경계선은 두께를 얇게 주면 들어가거나 나온다는 느낌을 거의 주지 못하기때문에 경계선두께를 굵게 하는것이 좋다. double도 마찬가지로 너무 얇게 주면 경계선이 2개로 보이지 않는다.

- 경계선색갈조절(border-color)

경계선의 색갈도 조절할수 있다.

기본형식은 《border-color:색갈명》이다.

색갈명은 "red" 등의 영문으로 입력해도 되고 "ff0000"같이 써도 된다.

실례:

```
<html>
<head>
<title>CSS 례제</title>
<style type="text/css">
font {border-color:orange;border-width:2;border-
style:solid}
</style>
</head>
<body>
<font>나는 할수 있다</font>
</body>
```

```
</html>
```

- 기타 선택항목

경계선두께와 관련된 선택항목에는 《border-방향-width:값》형식으로 된 선택항목도 있다. 실례로 아래경계선을 10px로 하고 싶으면《border-bottom-width:10》으로 하면 된다.

경계선모양과 관계된 선택항목에는 《border-방향-style:값》형식으로 된 선택항목도 있다. 실례로 오른쪽경계선만 점선으로 하고 싶으면 《border-right-style: dashed》로 하면 된다.

경계선색갈과 관계된 선택항목에는 《border-방향-color:값》이 있다. 실례로 우에 있는 경계선만 검은색으로 하고 싶다면 《border-top-color:black》로 하면 된다.

제5절. 려 파 기

PhotoShop에는 여러가지 효과를 줄수 있는 기능이 많다. CSS는 PhotoShop을 쓰지 않고도 그러한 효과를 가능하게 해준다. 물론 PhotoShop만큼 다양하고 세밀한 조정은 불가능하지만 CSS만으로도 일정한 정도의 효과는 실현할수 있다.

형식은 다음과 같다.

filter:filter 값(부분선택항목:값)

려파기조정은 filter외에 더 다른 선택항목은 없고 단지 filter선택항목에 여러개의 부분선택항목이 있다. 표기방식에서 본것처럼 부분선택항목은 선택항목뒤에 괄호를 넣어서 써주면 된다.

실례:

```
<html>
<head>
<title>CSS 레제</title>
<style type='text/css'>
image {filter:alpha(Opacity:100,style:2,finish Opacity:3)}
</style>
</head>
<body>
<image src=suhyun.gif>
</body>
</html>
```

alpha라는 려파값을 쓰고 Opacity라는 부분선택항목(filter의 속성이라고도 한다.)을 10으로 주려면 《filter:alpha(Opacity:10)》으로 쓰면 된다.

부분선택항목을 여러개 써줄 때는 구분기호로 반점(,)을 리용한다.

려파기조정은 려파기의 특성상 매몰방식을 쓰는 경우가 별로 없으므로 직접방식으로 표기한다.

- alpha효과

alpha효과는 어떤 객체의 투명도를 조절하는 기능을 수행한다.

실례:

```
<font style="height:5;filter:alpha
(Opacity:10,style:1,finishOpacity:100)">alpha효과
</font>
```

실례:

```
<image src=kinoko.jpg style="filter:alpha
(Opacity:10,style:1,finishOpacity:100)">
```

그림에 려파효과를 줄 때에는 그림자체에 따로 width나 height선택항목을 주지 않아도 filter효과가 나타나지만 글자는 반드시 범위를 잡아주어야 한다.

- 투명도 조절(Opacity, FinishOpacity)

Opacity는 alpha려파의 속성값으로 쓰인다. Opacity는 적용대상의 시작위치의 투명도값이고 Finish Opacity는 마지막위치에서의 투명도값이다. 그 값은 각각 0~100까지가 있다. 0을 준다면 가장 투명하게 되는것이고 100을 준다면 가장 불투명하게 된다.

실례:

```
<font
style="height:5;filter:alpha(opacity:10,finishopacity:100,style:1)">내용
</font>
```

이것은 시작되는 지점의 불투명도는 10으로 비교적 투명하게 나오고 끝나는 지점의 불투명도는 100으로 완전 불투명이 된다.

- 투명한 모양을 조절하는 style속성값

style속성은 투명도변화를 줄수 있는 모양을 지정해주는 기능을 수행한다.

style값은 0~3까지 줄수 있다. 0은 처음의 opacity값을 끝까지 유지하는 값이다. 1을 주면 왼쪽에서부터 오른쪽으로 가면서 opacity값에서 finishopacity값으로 투명도변화를 준다. 2는 중심의 opacity값에서 원형으로 바깥쪽으로는 finishopacity값으로 변화를 준다. 3은 x자 대각선의 opacity값에서 끝부분 변의 finishopacity값으로 변화를 준다.

실례:

```
<image src=kinoko.jpg
style="filter:alpha(Opacity:5,style:0,finishOpacity:100)">
<image src=kinoko.jpg
style="filter:alpha(Opacity:5,style:1,finishOpacity:100)">
<image src=kinoko.jpg
style="filter:alpha(Opacity:5,style:2,finishOpacity:100)">
<image src=kinoko.jpg
style="filter:alpha(Opacity:5,style:3,finishOpacity:100)">
```

그림가장자리를 뽀얗게 하려면 style값을 2로 주고 Opacity값은 크게, finish Opacity값은 작게 주면 된다.

실례:

```
<image src=kinoko.jpg    style="filter:alpha
(Opacity:100,style:2,finishOpacity:3)">alpha효과</font>
```

제 3 장. JavaScript 언어

제1절. JavaScript의 기초

JavaScript는 Netscape회사와 Sun Microsoft system회사가 공동으로 개발한 스크립트언어이다.

JavaScript가 열람기에 적재되기 시작한것은 Netscape2.0부터이다.

JavaScript는 다음과 같은 특징을 가지고있다.

- 스크립트언어이기때문에 콤파일할 필요가 없다.
- HTML문서안에 직접 JavaScript의 원천코드를 서술할수 있다.
- JavaScript대응의 열람기로부터 파일을 읽어넣으면 스크립트를 실행할수 있다.

즉 JavaScript는 HTML과 같이 특별한 개발환경이 필요없으며 word를 비롯한 문서편집기에서도 쉽게 프로그람을 작성할수 있다. 그리고 JavaScript로 작성한 프로그람은 대응한 열람기이라면 응용프로그람이나 조작체계가 달라도 꼭같이 동작시킬수 있다.

- JavaScript는 여러가지 조작체계의 열람기에서 실행할수 있다.
- HTML로는 실현하기 힘든 동적인 효과도 나타낼수 있다.

JavaScript를 사용하면 지금까지 CGI로서 진행하고있었던 처리의 일부나 CGI로써는 할수 없었던 처리를 웨브페지상에서 진행할수 있다. 례를 들면 상품의 주문이나 등록정보를 받아들이는 웨브페지에서는 우선 사용자가 요구하는 제품명이나 수량 등의 정보를 홈페지에 입력하거나 선택한 다음 그것들의 정보를 기초로 자료를 묶고 계산을 진행하며 금액의 총합 등의 결과정보를 페지에 현시한다.

이러한 일련의 처리를 JavaScript를 사용하지 않고 진행하는 경우에는 일반사용자로부터 망을 통하여 CGI가 자료를 받아 계산한 결과를 다시 봉사기로부터 망을 통하여 사용자에게 보내는 절차가 필요하다.

그렇게 되면 망의 상태나 봉사기와의 대화상태에 따라서 CGI로부터의 응답이 잘 되지 않을수 있는 등 처리시간이 길어지는 문제가 발생함으로 효률적이라고 말할수 없다.

그러나 JavaScript를 사용하면 이 문제는 해결된다.

JavaScript는 홈페지의 자료를 보내기 전에 홈페지의 입력항목의 검사를 진행하거나 자료를 CGI에 보내지 않고 열람기상에서 계산을 진행할수 있다. 결국 망에서의 통신시간을 줄일수 있고 순간에 사용자에게 회답을 되돌려줄수 있다.

JavaScript는 HTML로서 표현할수 없는 세밀하고 움직임을 주는 효과도 웨브페지상에서 실현할수 있다. 례를 들어 HTML에서는 창문을 열고 싶을 때에 open객체를 사용한다. 그러나 open객체에서 열기창문을 페지작성자가 조정할수 없다.

JavaScript를 사용하면 창문을 여는것만이 아니라 높이나 너비, 안내띠나 도구띠의

유무, 그리고 화면상에서 표시위치 등을 구체적으로 조정할수 있다. 또한 HTML에서는 여러개의 프레임을 한번밖에 변경할수 없지만 JavaScript를 사용하면 가능하다.

이와 같이 JavaScript로는 HTML만으로서는 실현할수 없는것도 웨브페지상에서 실현할수 있다.

3.1.1. JavaScript의 서술방식

JavaScript를 서술하는 방법에는 2가지가 있다.

HTML파일안에 서술하는 방법과 스크립트만으로 서술한 파일을 HTML파일이 호출하는 방법이 있다.

JavaScript는 HTML파일안에 서술하는 언어이므로 꼬리표를 리용한다.

이 절에서는 스크립트를 작성할 때 반드시 필요한 꼬리표에 대하여 해설한다.

- HTML파일에 서술하는 방법

HTML파일에 JavaScript를 서술할 때에는 <script>～</script> 꼬리표를 사용한다. 그리고 <script>꼬리표의 language객체에 JavaScript라고 지정한다.

실례:

```
<script language="JavaScript">
<!--
document.write("여기는 JavaScript부분이다.")
-->
</script>
```

JavaScript는 대부분 <head>와 </head>사이에 삽입한다. 여기서 language="JavaScript"부분은 생략해도 된다.

열람기는 자기가 지원하지 않는 꼬리표를 무시한다. 따라서 자기가 지원하지 않는 원천코드내용이 그대로 공개된다. 이것을 방지하기 위해 해설문을 넣는다.

JavaScript도 꼬리표와 마찬가지로 문서의 우측부터 차례대로 처리된다.

따라서 어디에 JavaScript원천을 넣는가에 따라 JavaScript가 언제 실행되는가가 결정된다.

- JavaScript를 외부문서에서 호출하는 방법

많은 문서를 만들 때 공통되는 JavaScript원천이 있다면 그것을 일일히 문서마다 서술하자면 상당한 시간과 노력이 필요된다. 이것은 JavaScript원천을 파일로 따로 만들어 문서를 작성할 때 간단하게 련결시키는 방법으로 실현할수 있다.

형식은 다음과 같다.

<script src= “스크립트파일의 url” ></script>

스크립트파일에는 JavaScript의 원천코드만을 서술한다.

그리고 이 스크립트파일의 확장자는 js로 한다. 례를 들면 JavaScript의 원천코드를

Script.js라는 파일에 서술하고 HTML파일과 같은 등록부에 보존하였다고 하자. 이 경우 HTML파일에서는 다음과 같이 서술한다.

<script src= "Script.js" ></script>

우와 같은 형식으로 준다면 수십개의 문서에다가 간단한 명령 하나만으로 똑같은 JavaScript원천을 한번에 줄수 있다는 우점이 있다. 다만 다른 파일에 적재하므로 문서자체를 적재하는 시간은 약간 길어질수 있다.

3.1.2. JavaScript의 구조

1) 객체

JavaScript의 객체란 JavaScrip실행을 위한 목적, 적용대상을 말한다.

JavaScript객체는 크게 내장객체와 조립객체로 나눈다.

객체들의 중요한 특징은 다음과 같다.

- 내장객체

내장객체는 열람기자체가 가지고있는 정보나 열람기의 매개 부분품을 표시하는 객체이다.

내장객체에는 차림표띠 등의 부분품, 대화칸(form), 그림외에 HTML로서 표시되는 부분, 사용자정보 등을 표시하는 여러가지 종류의 객체가 있다. 내장객체사이에는 계층관계가 존재한다.

서술할 때에는 계층관계를 정확하게 지정해야 한다. 례를 들면 image객체를 사용할 때에는 window.document.image라고 계층관계에 따라서 서술하여야 한다. 이 경우의 서술방법은 객체와 객체의 사이를 《.》으로 구별하여 차례로 window. document이라고 련결한다.

image객체인 경우 document객체로부터 하나 아래계층의 객체이므로 window. document.image라고 서술한다.

최상위객체는 창문자체를 표현하는 window이고 그 아래에 document나 location 등의 하위객체들이 있다. 례를 들어 window.document.write()라고 하면 window -> document의 순서로 세부적인 객체에 접근하는것이다. 그러나 window객체는 대부분의 경우 생략한다.

객체의 계층관계는 다음과 같다.

표 3-1. 객체의 계층관계

상위 ◁			▷ 하위
	document	image	text
window	location	form	hidden
	frame	applet	reset

	history	link	radio
		area	href
		layer	select
			fileupload
			password
			textarea
			submit
			button

- 조립객체

조립객체는 JavaScript에 있는 날자를 취급하거나 원주률이나 시누스값 등을 계산하기 위한 객체를 말한다.

- date객체

date객체는 날자나 시간을 얻거나 설정을 진행하는 객체이다. date객체는 호출사용자가 사용하고있는 콤퓨터의 국부시간정보를 사용하므로 CGI에서는 힘들었다. 이것을 리용하여 세계각지의 국부시간에 맞추는 처리를 진행할수 있다. date객체는 new연산자를 사용하여 객체를 작성하고 그 객체에 대하여 처리를 진행한다.

- math객체

math객체는 수학연산을 할수 있는 객체이다. math객체는 math의 뒤에 속성이나 메쏘드를 서술하여 사용한다. 이 객체를 리용함으로써 봉사기에 자료를 보내지 않고 사용자의 열람기만으로서 여러가지 계산을 진행할수 있다.

- string객체

string객체는 문자렬에 대하여 수정이나 검색 등의 조작을 진행하는 객체이다. JavaScript1.1부터 new연산자를 사용하여 객체를 작성할수 있게 되였다.

- array객체

array객체는 배렬의 작성이나 조작을 진행하는 객체이다.

2) 속성

JavaScript에서 속성이란 특정한 실체가 가지는 특징을 말한다. 그러므로 객체와 함께 쓰인다. 속성에는 읽기전용인것과 변경가능한것이 있다.

- 읽기전용속성

읽기전용속성에는 내장객체의 appCodeName속성과 appName속성이 있다. 또한 screen객체의 width속성, height속성 등도 사용자가 변경할수 없는 읽기전용속성이다. 이 속성들은 다 사용자가 열람기환경에 맞는 값을 준다. 이 값을 스크립트에서 잘 리용하면 호출사용자의 매 환경에서 최적상태의 웨브페지를 표시할수 있다.

- 변경가능한 속성

변경가능한 속성에는 location객체의 href속성이나 image객체의 src속성들이 있다. 또한 홈안의 값을 표시하는 text객체의 value속성들이 있다.

이 변경가능한 속성값들은 우에서 서술하는 사건처리에 의하여 동적으로 변경된다.

3) 메쏘드

메쏘드(Method)란 객체의 동작을 나타내는 속성이다. 대표적인 메쏘드로는 write()가 있는데 이것은 내용을 출력할 때 리용하는 메쏘드이다. 여기서 괄호안에는 메쏘드의 구체적인 선택항목이나 내용을 입력한다.

JavaScript에는 많은 메쏘드가 있다.

실례:

```
<html>
<head>
<title>JavaScript 례제페지</title>
<script language="JavaScript">
document.write('나는 할수 있다')
</script>
</head>
<body>
</body>
</html>
```

4) 함수

함수란 어떤 특정한 처리를 묶어놓은것으로서 메쏘드와 비슷하지만 객체와 붙여서 쓰지 않는다는 점에서 차이가 있다.

document.write()처럼 메쏘드는 객체와 함께 쓰이지만 함수는 좀 다르다. 또한함수도 객체와 마찬가지로 사용자자신이 직접 정의하는 함수가 있고 원래부터 있던 내장함수가 있다.

함수는 메쏘드와 마찬가지로 괄호()를 사용하며 이 속에 문자렬이나 변수를 넣을수 있다.

함수이름은 일정한 조건을 붙인것과 사용자측에서 임의로 정의할수 있다.

어떤 조건에 대하여 진행하고싶은 처리나 자주 반복되는 처리는 함수로 정의한다.

함수를 선언하는 방식은 다음과 같다.

```
function 함수이름(인수,인수,…){
          처리내용}
```

함수의 정의는 <head> ∼</head>안에서 진행하며 함수의 호출은 <body>∼<body>안에서 진행한다.

실례:

```
<html>
<head>
<title>JavaScript례제페지</title>
<script language="JavaScript">
function abcd(){document.write('훌륭하다.~ ^^')}
</script>
</head>
<body>
<a href="javascript:abcd()">찰칵!!</a>
</body>
</html>
```

5) 사건처리

JavaScript에서는 마우스가 눌리웠거나 특정한 동작이나 상태가 발생하였을 때를 사건이 발생하였다고 말한다.

어떤 사건이 발생하였을 때 그 사건을 얻거나 특정한 스크립트를 호출하는것이 사건처리이다.

사건처리를 사용하면 여러가지 동작에 따라 다른 동작을 지정할수 있으므로 HTML만으로는 표현할수 없었던 보다 동적인 효과를 실현할수 있다.

사건처리에서는 설정할수 있는 객체가 정해져있다.

- onClick

onClick는 객체를 눌렀을 때의 사건처리를 얻는다. 례를 들면 button객체에 onClick를 설정하였다고 하면 홈의 단추를 눌렀을 때에 새로운 창문을 열거나 다른 페지를 표시하는 기능을 실현할수 있다.

onClick는 button객체, checkbox객체, link객체, radio객체, reset객체, submit객체에 설정할수 있다.

- onLoad

onLoad는 페지나 그림이 읽어질 때의 사건처리를 얻는다. 페지를 읽을 때에 상태행이나 홈페지의 문자를 움직이거나 그림을 움직이는 등 페지가 읽어지면 동시에 처리를 진행하도록 하는 경우에 리용한다.

onLoad는 JavaScript1.0에서는 window객체, JavaScript1.1에서는 image객체에 설정할수 있다.

페지를 읽으려는 사건을 취급할 때에는 <body>안에 onLoad를 설정한다.

- onSubmit

onSubmit는 홈의 onSubmit단추를 눌렀을 때의 사건을 취급한다. 대화칸의 자료가 보내지기 전에 사건을 취급하므로 처리가 끝날 때까지 자료는 전송되지 않는다. 스크립트실행시에 자료를 보내는 처리를 중지할수 있으므로 대화칸의 내용을 검사하고 문제가 있는 경우 사용자에게 경고하여 송신처리를 중지할수 있다. onSubmit는 form객체에 설정할수 있다.

- onChange

onChange는 현재의 자료가 이전 자료와 바뀌여 졌을 때 발생하는 사건을 처리한다. onChange를 select객체에 설정하면 대화칸의 선택항목이 변경되였을 때에 처리를 얻어 대화칸안의 값을 계산할수 있다. 또한 변경하고싶지 않은 대화칸의 내용이 변경 되였을 때에 사용자에게 경고하는것과 같은 처리를 진행할수 있다. onChange는 JavaScript1.0에서는 select객체, text객체, textarea객체, JavaScript 1.1에서는 fileupload객체에 설정할수 있다.

- onMouseOver

onMouseOver는 객체우에 마우스유표를 놓았을 때의 처리를 취급한다. onMouseOver를 link객체에 설정하면 마우스유표가 련결상에 놓일 때에 상태행에 련결에 관한 설명을 표시하거나 그림을 바꾸거나 하는 처리를 진행할수 있다. onMouseOver는 area객체, link객체에 설정할수 있다.

- onMouseOut

onMouseOut는 마우스유표가 객체에서 떨어졌을 때의 처리를 취급한다. link객체에 설정하면 마우스유표가 련결상에서 떨어졌을 때에 onMouseOver를 사용하여 바뀐 그림을 원래대로 보내는 처리를 진행할수 있다. JavaScript1.1에 추가된 사건처리이며 area객체, link객체에 설정할수 있다.

제2절. JavaScript의 주요객체

3.2.1. JavaScript document 객체

- document객체의 속성

document객체의 속성(document.속성)을 리용하여 문서와 관련된 여러가지 본문이나 배경, 그림을 조정할수 있다.

일반형식은 다음과 같다.

```
객체.속성이름="값"
```

① 배경속성 bgColor

문서(document)의 배경을 지정하는 속성이다.

인용부호를 쓰는가, 안쓰는가에 따라 의미가 달라진다.

실례:

document.bgColor="red"(document.bgColor속성값을 red로 한다는 뜻)

document.bgColor=red (document.bgColor를 red라는 변수로 정의한다는 뜻)

값에는 영문이름도 되고 코드이름도 가능하다.

실례:

```
<script language="JavaScript">
document.bgColor="#ff0000"
</script>
```

② 서체색갈 fgColor

서체색갈과 관련된 속성이다.

실례:

```
<script language="JavaScript">
document.fgColor="blue"
</script>
```

③ 련결색갈 linkColor, alinkColor, vlinkColor

련결할 때 서체색갈을 조절할수 있는 속성이다. linkColor는 방문전의 련결색갈이고 alinkColor는 마우스를 찰칵했을 때의 련결색갈, vlinkColor는 방문후의 련결색갈이다.

실례:

```
<script language="JavaScript">
document.linkColor="green"
```

```
document.alinkColor="red"
document.vlinkColor="violet"
</script>
<a href=http://www.univ.co.kp target="_blank">찰칵</a>
```

④ 페지이름 title

페지의 이름과 관련된 속성이다. 열람기의 우측 푸른색부분에 있는 제목을 말한다.

실례:

```
<script language="JavaScript">
document.title="살기좋은 내나라"
</script>
```

⑤ 문서의 주소관련속성 url

문서의 주소와 관련된 속성이다. 물론 다른 속성처럼 《document.url="주소이름"》을 써서 값을 지정해줄수 있지만 이미 주소가 있는 문서의 주소를 다시 그렇게 직접 써줄 필요가 없다. 그러므로 속성자체의 정의보다는 write()메쏘드를 써서 해당 문서의 주소를 알아보도록 하는데 많이 사용하고있다.

실례:

```
<script language="JavaScript">
document.write(document.url)
</script>
```

⑥ 마지막 수정날자 lastModified

마지막 수정날자와 관련된 속성이다. 이것 또한 사용자자신이 직접 마지막 수정날자를 정의해줄 필요가 없으므로 확인의 용도로만 쓰이고있다. 즉 마지막 수정날자를 알려고 할 때 리용한다.

실례:

```
<script language="JavaScript">
document.write(document.lastModified)
</script>
```

⑦ 쿠키관련 cookie

쿠키와 관련된 속성이다.

실례:

```
<script language="JavaScript">
document.write(document.cookie)
</script>
```

⑧ 그밖의 속성들

문서의 form전송과 관련된 forms속성, 문서의 그림과 관련된 images속성, 련결과 관련된 anchors, 삽입과 관련된 embeds, 령역과 관련된 domain, java애플레트와 관련된 applets속성 등이 있다.

실례:

```
<html>
<head>
<title>실례</title>
<script language="JavaScript">
document.bgColor="yellow"
document.fgColor="blue"
document.linkColor="green"
document.alinkColor="red"
document.vlinkColor="violet"
document.title="살기좋은 내나라"
document.write(document.URL)
document.write(document.lastModified)
</script>
</head>
<body>
<a href=http://univ.co.kr target="_blank"> 평양
</body>
</html>
```

- document객체의 메쏘드

① write메쏘드

문서출력과 관계되는 메쏘드인데 작성한 문서를 직접적으로 보여준다.

형식은 다음과 같다.

document.write("내용")

실례:

```
<script language="JavaScript">
<!--
document.write("안녕하십니까?")
-->
```

```
</script>
```

여기서 열람기는 될수록이면 완성된 형태의 HTML문서를 요구하기때문에 JavaScript원천만 넣기보다도 아래와 같이 HTML문서형식안에 JavaScript원천을 넣어 사용하여야 한다. 물론 JavaScript원천만 넣어도 열람기는 최대한 실행시키지만 복잡한 코드나 <body>의 onload를 원하는것들은 HTML문법을 지켜야 제대로 출력이 되므로 될수록 HTML문서형식을 모두 쓰는 습관을 가져야 한다.

실례:

```
<html>
<head>
<title>제목</title>
</head>
<body>
본문내용
</body>
</html>
```

② 꼬리표도 사용가능

《document.write("")》의 인용부호사이에는 꼬리표를 넣어도 된다. 즉 하나의 작은 HTML문서를 삽입하는것으로 된다. 또한 HTML문서에 쓸수 있는 모든 꼬리표뿐만 아니라 CSS를 인용부호사이에 넣어도 된다. 그러나 JavaScript구문을 넣을 때는 인용부호를 넣지 말아야 한다.

```
<script language="JavaScript">
<!--
 document.write("<marquee>안녕하십니까?</marquee>")
 -->
</script>
```

JavaScript괄호안의 《("내용")》과 《(내용)》에는 차이가 있다. 첫번째는 인용부호안에 있는 문자를 출력하라는 뜻이고 두번째는 내용이라는 글자자체를 변수로 취급하라는 뜻이다. 두번째 경우는 《내용》이라는 변수를 정의하는 다른 구문을 필요로 한다. 변수를 정의하는것을 선언이라고 한다. 선언에는 객체의 선언, 메쏘드의 선언 등이 있다. 따라서 앞의 례제와 같이 단순한 출력만을 할 때에는 인용부호를 넣어주는것이 옳은 표현이다.(작은 인용부호도 가능)

괄호안에 JavaScript구문을 넣고 거기에 인용부호를 넣어주면 그것자체를 그냥 출력한다.

실례:

("안녕하십니까?") (◯)

('안녕하십니까?') (◯)

※ 작은 인용부호도 허용됨

(document.write())

이것은 《document.write()》를 처리하라는것을 의미한다.

("document.write()")

이것은 《document.write()》라는 글자를 출력하라는것을 의미한다.

두개이상의 구문을 처리할 때 구문마다 줄바꾸기를 하거나 한줄로 처리할 때 구문의 끝에 반두점(;)을 붙인다.

실례:

```
document.write("안녕하십니까?");document.write("반갑습니다~~!")
                    //반두점을 붙임
document.write("안녕하십니까?")
document.write("반갑습니다~~!")//한줄을 띄움
```

③ 속성이나 변수 등의 값을 표시

다른 구문에서 정의해준 여러 객체나 함수, 메쏘드들의 값도 알기 쉽게 표시해준다. 이때 인용부호는 생략한다.

아래의 례제는 먼저 배경속성을 red로 설정해주고 다음 줄에서 write메쏘드를 사용하여 배경색을 표시해주는 간단한 례제이다.

실례:

```
<script language="JavaScript">
<!--
document.bgColor="red"
document.write(document.bgColor)
-->
</script>
```

bgColor의 속성값을 JavaScript의 document.write로 확인하는 례제이다. 괄호안의 내용에 여러가지 기능을 동시에 입력하려는 경우에는 《+》연산자를 사용한다. 례를 들어 우의 례제에서 현시내용과 속성을 동시에 현시하려면

```
<script language="JavaScript">
<!--
document.bgColor="red"
document.write("배경색은..." + document.bgColor +"이다.")-->
```

```
</script>
```

속성을 정의할 때에는 인용부호를 쓴다.

실례:

document.bgColor="red" // 문서의 배경색은 red이다.

인용부호를 쓰지 않으면 속성자체를 하나의 변수로 간주한다.

실례:

document.bgColor=red //document.bgColor자체를 red라는 변수로 간주한다.

괄호안에 인용부호를 쓰지 않는 경우는 JavaScript구문을 쓰거나 속성을 쓸 때, 함수의 값을 정의할 때 등이다.

실례:

document.write(document.bgColor) // document.bgColor자체를 출력하는것이 아니라 document.bgColor속성값을 출력한다.

실례:

history.back(1) // back의 값을 1로 한다.

실례:

function abcd(ok) // abcd라는 함수가 있는데 괄호안의 값을 ok라는 변수로 지정한다.

3.2.2. JavaScript history 객체

History라는것은 력사라는 말이나 지나간 사실을 말한다. 웨브페지를 열람할 때에도 지나간 사실들이 있다. 바로 앞페지에서 왔다던가 다른데서 련결되여 왔을 때 그 페지들은 전부 다 지나간 사실이라고 할수 있다. history객체는 바로 이런 페지들로 되돌아가거나 앞으로 갈 때 리용하는 명령이다. 간단하게 말해서 웨브열람기의 《뒤로》, 《앞으로》기능과 같다.

history에는 3가지의 메쏘드, 즉 이전 페지로 돌아가는 back()메쏘드, 이전 페지에서 다시 앞으로 돌아올 때 쓰는 forward()메쏘드, 우의 두 메쏘드의 개념을 통합한 go()메쏘드들이 있다

- back()메쏘드

실례:

```
<script language="JavaScript">
<!--
history.back(1)
-->
</script>
```

해당 페지를 열면 무조건 이전 페지로 돌아간다. 괄호안에 수자를 넣으면 수자만큼 이전 페지로 간다. 3을 주면 3번째 전페지로 간다.

- forward()메쏘드

실례:

```
<script language="JavaScript">
<!--
history.forward(1)
-->
</script>
```

이전 페지로 갔을 때 앞의 페지로 다시 돌아가는 역할을 한다. 따라서 뒤로가기를 하지 않은 경우라면 아무런 변화도 일어나지 않는다. 물론 여기서도 괄호안에 수자를 넣으면 수자만큼 앞으로 간다.

- go()메쏘드

실례:

```
<script language="JavaScript">
<!--
history.go(-1)
-->
</script>
```

go()메쏘드는 괄호안에 수자를 반드시 넣어주어야 한다. 괄호안에 넣는 수자로서는 옹근수를 넣으면 된다. 만일 -1을 넣으면 1만큼 과거로 돌아가고 1이라고 하면 1만큼 앞으로 나간다. 즉 부의 옹근수를 넣으면 back()와 같은 개념이 되고 자연수를 넣으면 forward()와 같은 개념이 된다.

- 함수와 조합

사실 history객체의 back(), forward(), go()메쏘드만 가지고 코드를 작성하지 않는다. 이 메쏘드만을 리용한다면 페지를 열기가 바쁘게 전페지로 돌아가거나 앞페지로 이동하기때문에 페지를 열람할수 없기때문이다. 그래서 단추를 만들어서 그것을 찰칵하면 뒤로가기를 한다든가 하는 상태를 만들기 위하여 이 메쏘드들과 함수들을 조합하여 리용하고있다. 이 메쏘드를 함수속에 포함시키면 함수가 대응되였을 때 실행이 된다.

형식은 다음과 같다.

```
function 함수이름(){JavaScript 구문}
```

함수이름을 abcd라고 정한다면 abcd()가 실행되였을 때 중괄호({})안의 구문이

실행되므로 이 메쏘드를 쓰는것이 편리하다. back()메쏘드를 abcd()라는 함수안에 넣으면 《function abcd(){history.back()}》 와 같이 된다.

함수를 호출하는 방법은 여러가지가 있으나 우와 같은 경우에는 단추를 눌러서 호출하는 방법으로 하는것이 좋다.

<a>꼬리표를 리용하는 경우

실례:

```
<script language="JavaScript">
<!--
function abcd(){history.back(1)}
-->
</script>
<a href="javascript:abcd()">뒤로가기</a>
```

<form> 와 <input>를 조합하는 경우

실례:

```
<script language="JavaScript">
<!--
function abcd(){history.back(1)}
-->
</script>
<form action="javascript:abcd()">
<input type=submit value="찰칵">
</form>
```

<a>꼬리표의 href와 <form>꼬리표의 action에 javascript:abcd()를 넣음으로써 찰칵하면 abcd라는 함수를 실행하게 만들었다. JavaScript에는 <a>나 <form>꼬리표가 아니라도 마우스찰칵을 대신할수 있는것이 있다. 바로 onclick라는 사건호출기이다.

실례:

```
<script language="JavaScript">
<!--
function abcd(){history.back(1)}
-->
</script>
<textarea rows=4 cols=30 onclick="abcd()">본문공간이다.</textarea>
```

<textarea>꼬리표는 본문칸을 만든다. 그러나 실례에서는 여기에 onclick를 사용함으로써 본문을 입력할수 있게 하는것이 아니라 이것자체를 단추화하였다. 따라서 그 공간을 찰칵하면 abcd()라는 함수가 호출된다.

3.2.3. JavaScript window 객체

window객체라는것은 하나의 폐지자체에 대응하는 객체로 생각하면 된다. 앞에서 언급한것처럼 이 객체는 모든 객체의 최상위객체로 된다. 그 하위객체는 특정한 령역(문서라든가, 지나간 사실, 련결 등)을 전문으로 담당하는 반면에 window객체는 그것들을 다 포함하는 하나의 완결된 폐지를 목표로 한다. 따라서 모든 하위객체는 원칙적으로 window객체를 써준 다음에 자신의 객체를 써주어야 한다.

실례로 document객체의 경우에 원래는 window.document이고 history객체도 원칙적으로는 window.history이다. 하지만 JavaScript는 폐지를 목표로 하는 언어이므로 window라는 객체를 생략해도 된다.

window도 다른 객체와 마찬가지로 많은 속성과 메쏘드가 있다. 례를 들어 창문을 강제로 여는 open()메쏘드는 window객체의 메쏘드이므로 window.open()이지만 앞서 말한바와 같이 window는 생략해도 되므로 open()자체만 써도 된다는것이다.

만일 window객체를 생략해버리면 JavaScript를 배우려는 경우 어떤것에는 객체가 있고 어떤것에는 없으므로 메쏘드 또는 속성이라든가 하는 개념을 가지는것이 힘들게 된다.

객체가 씌여져있지 않은것은 최상위 객체인 window를 생략한 경우이거나 아니면 함수이다.

- JavaScript window객체의 속성

속성은 《속성이름="정의값"》의 형식으로 정의해준다. 인용부호를 넣지 않으면 정의값자체를 변수로 취급하므로 정의값을 따로 정의해주어야 한다.

window객체의 속성은 창문전체에 관련된 여러가지 상태띠나 창문의 이름값 등에 대한 설정을 한다. 말그대로 폐지자체의 속성을 정해주는 역할을 한다. 물론 window객체는 생략할수도 있다.

① 상태띠 기정통보문 defaultStatus

상태띠는 창문하단의 작은 본문창을 말한다. 이것은 HTML꼬리표로는 표현할수 없다.

window객체의 defaultStatus속성을 리용하면 상태띠에 원하는 문자를 넣을수 있다.

실례:

```
<script language="JavaScript">
window.defaultStatus="나는 할수 있다"
</script>
```

우의 코드를 실행한 다음 상태띠를 보면 《나는 할수 있다》의 문자가 출력된것을 알수 있다.

defaultStatus속성으로 설정한 본문은 상태띠의 기정통보문으로 되므로 하이퍼련결에 마우스를 올려놓으면 URL이 표시되였다가도 마우스를 떼면 다시 설정된 본문이 나타나게 된다.

② 상태띠통보문 status

상태띠의 통보문을 줄수 있는 속성이다. 주로 사건조종과 같이 쓰이는데 간단한 실례를 보기로 하자.

실례:

```
<script language="JavaScript">
function abcd(){window.status="내나라 제일로 좋아! "}
</script>
<a        href=http://www.univ.co.kp        onmouseover="abcd();return
true">내나라! </a>
```

함수를 실행하는 방법은 다음과 같다.

function 함수이름(){JavaScript구문}

함수이름을 abcd라고 하면 abcd()가 실행되였을 때 중괄호 {}안의 구문이 실행된다. 중괄호안의 내용은 또 다른 작은 HTML문서라고 생각하면 된다. 거기에는 꼬리표, CSS, JavaScript의 구문이 들어갈수 있다.

③ 기타 속성들

windows: 현재 활성화되여있는 창문을 말한다. 창문이름을 지정할수도 있다.

name: 창문이름을 지정한다. windows와 같다.

self: 현재 사용중인 객체를 지정한다.

opener: open()메쏘드로 열린 창문의 이름을 지정한다. 이름을 지정하는 리유는 마지막에 목표지정을 편리하게 하기 위해서이다.

frames: window객체가 가진 프레임에 대한 정보를 나타낸다.

length: 프레임을 분할하였을 때의 프레임의 개수를 나타낸다.

- JavaScript window객체의 메쏘드

① open()메쏘드

새로운 창문을 여는 메쏘드이다.

형식은 다음과 같다.

```
window.open("새로 열게 될 주소",
     "열게 될 문서의 이름","기타 선택항목")
```

window객체의 메쏘드이므로 window.open() 또는 open()으로 써준다.

괄호안에 아무 내용을 넣지 않으면 단순한 빈창문을 열게 되고 URL을 넣으면 해당주소로 이동하게 된다. 이때 인용부호를 반드시 써주어야 한다.

실례:

```
<script language="JavaScript">
<!--
window.open("http://www.univ.co.kp")
-->
</script>
```

실례:

```
<script language="JavaScript">
<!--
window.open("http://www.univ.co.kp","univ","width=150,height
=150")
-->
</script>
```

두번째 인용부호의 "univ"에서

- univ라는 이름을 가진 창이 이미 있을 경우에는 거기에서 《내나라》홈페지로 이동한다.
- univ라는 창이 없을 경우에는 새 창을 열고 《내나라》홈페지를 현시한다.
- 련결을 할 때 target="small"로 하면 그 창에서 련결이 된다는 뜻이다.

세번째 인용부호의 "기타 선택항목"에는 다음과 같은것이 있다.

- Menubar(yes/no)-안내띠의 현시유무를 설정한다.
- Toolbar(yes/no)-도구띠의 현시유무를 설정한다.
- Location(yes/no)-주소표시칸의 현시유무를 설정한다.
- Status(yes/no)-상태표시줄의 현시유무를 설정한다.
- Scrollbar(yes/no/auto)-흘림띠의 현시유무를 설정한다.
- Resizable(yes/no)-크기를 조정할수 있는가를 설정한다.
- Width(수자, 단위는 px)-너비
- Height(수자, 단위는 px)-높이

실례:

window.open("http://www.univ.co.kp","univ","width=150,height=150,scrollbars=yes")라고 한다면 《http://www.univ.co.kp》라는 주소를 가진곳을 가로 150px와 세로 150px, 흘림띠는 주고 univ의 이름으로 띄우라는 명령이다.

width=150, height=150처럼 매 선택항목들사이에는 반점(,)을 넣는다.

강제로 띄우지 않고 무엇인가를 찰칵하여 띄우게 할수 있다. 이것을 실현하자면 open()메쏘드를 함수속에 포함시키면 된다. 그리고 함수가 호출될 때에만 open()메쏘드가 실행되도록 하면 된다. 물론 함수도 강제로 호출이 되는 경우가 아니라 찰칵을 한다든지 기타 다른 메쏘드로 호출되도록 해야 한다. 이렇게 하는것을 사건조종이라고 한다.

실례:

```
<script language="JavaScript">
<!--
function
abcd(){window.open("http://www.univ.co.kp","univ","width=150
,height=150")}
-->
</script>
<a href="javascript:abcd()">내나라</a>
```

② close()메쏘드

open()과는 반대로 창문을 닫는 역할을 하는 메쏘드이다. 괄호안에는 내용을 넣어줄 필요가 없다.

실례:

```
<script language="JavaScript">
<!--
window.close()
-->
</script>
```

강제로 닫지 않고 무엇인가를 찰칵하는 방법으로 닫게 할수도 있다. 그것은 우에서 보여준것과 같은 사건조종원리를 리용하면 된다.

실례:

```
<script language="JavaScript">
<!--
function abcd(){window.close()}
-->
</script>
<a href="javascript:abcd()">창닫기</a>
```

③ alert()메쏘드

window객체의 alert메쏘드는 경고창이 나타나게 하는 역할을 한다.

그림 3-1. alert()메쏘드

alert메쏘드는 우의 그림과 같이 경고창이 나타나게 함으로써 작성자의 의견을 전달한다. 쌍방의 의견이 오가는 comfirm메쏘드나 prompt메쏘드와는 조금 다르다. 그러므로 alert()의 괄호안에는 현시내용만을 입력하면 된다.

실례:

```
<script language="JavaScript">
<!--
window.alert("나는 할수 있다")
-->
</script>
```

실행화면을 열자마자 경고창이 나타나는 데 이렇게 alert()는 첫화면부터 무조건적인 실행을 하기때문에 공개내용이나 주의사항을 나타낼 때 자주 리용된다.

경고창이 무턱대고 나타나지 않도록 하려면 사건조종을 리용하여야 한다. 즉 단추를 누르거나 마우스로 지적하여야 경고창이 나타나도록 하게 하는것이다.

실례:

```
<script language="JavaScript">
<!--
function abcd(){window.alert("나는 할수 있다")}
-->
</script>
<a href="javascript:abcd()">찰칵</a>
```

우의 실례를 실행시키고 찰칵하면 경고창이 뜬다.

실례:

```
<script language="JavaScript">
<!--
function abcd(){window.alert("나는 할수 있다")}
-->
</script>
```

```
<a href="javascript://"onmouseover=abcd()>마우스를 여기에</a>
```

실례는 마우스를 가져가기만 하면 경고창이 나타나도록 하는 사건조종을 리용하여 실현한것이다. 마우스를 가져가면 그림이 변하는것은 바로 이러한 원리를 리용한것들이다.

④ confirm()메쏘드

confirm메쏘드는 상대방의 의견을 물어보는 기능을 수행한다. alert메쏘드가 내용만을 전달한다면 confirm메쏘드는 선택의 가능성을 주는 메쏘드이다.

그림 3-2. confirm()메쏘드

confirm()의 괄호안에 현시내용을 넣는다.

실례:

```
<script language="JavaScript">
<!--
 window.confirm("나는 할수 있다")
-->
</script>
```

confirm자체만으로는 《확인》단추를 찰칵해도 《취소》단추를 찰칵해도 아무 반응이 없다. 그러므로 confirm은 특성상 if-else명령문과 같이 쓰이는 경우가 많다.

《if(){} else{}》명령문은 《만일 if괄호안의 값이라면 그다음에 있는 중괄호({})안의 내용을 수행하고 아니면 else의 중괄호({})안의 내용을 수행하라》는것이다. {}안에는 요구하는 구문을 넣으면 된다. 실례를 들어 《if(a==1){document.write("좋다")}else{document.Write ("싫다")}》명령문은 a의 값이 1이면 《좋다》라는 글자를 출력하고 1이 아니라면 《싫다》라는 글자를 출력하도록 한다.

실례:

```
<script language="JavaScript">
<!--
a=1
if(a==1){document.write("좋다")}else{document.write("싫다")}
```

```
        -->
        </script>
```

a=1이라고 미리 정해주므로 if문의 내용이 실행된다.

a=1은 var a=1이라고 써주어도 된다. 이것을 《선언》이라고 한다.

다음 a의 값을 선택에 따라 다르게 설정해보자.

실례:

```
<script language="JavaScript">
<!--
a=window.confirm("선택하시오")
if(a==true){document.write("좋다")}else{document.write("싫다")}
-->
</script>
```

바로 이런 경우에 confirm()메쏘드를 쓰는것이 편리하다. confirm()은 확인을 누르면 true로, 취소를 누르면 false로 인식한다. 이것을 a라는 변수로 확정하고 if-else구문을 리용하여 a가 true라면 즉 《확인》단추를 찰칵하면 《좋다》를 출력하고 그렇지 않으면 《싫다》를 출력한다.

실례:

```
<script language="JavaScript">
<!--
if(window.confirm("선택하시오")==true){document.write("좋다")}
else{document.write("싫다")}
-->
</script>
```

실례:

```
<script language="JavaScript">
<!--
if(window.confirm("학교로 가겠습니까?")==true){
window.open("http://www.univ.co.kp")
}
else{
window.alert("왜 가지 않습니까?")
}
-->
```

```
</script>
```

⑤ prompt()메쏘드

prompt메쏘드란 띠를 나타나게 하여 그 내용을 즉시 처리하려고할 때 쓰는 메쏘드이다. confirm메쏘드가 《확인》, 《취소》의 객관형식이였다면 prompt메쏘드는 임의의 문자를 넣을수 있는 주관형식이다.

그러므로 prompt메쏘드로는 입력부분에 문자를 미리 입력해놓을수 있다. 그것은 《prompt("현시내용", "입력창에 현시내용") 》과 같이 입력하면 된다.

실례:

```
<script language="JavaScript">
<!--
window.prompt("나는 할수 있다")
-->
</script>
```

prompt()의 기능은 단순히 입력창을 만들어주는것에 불과하다. write()메쏘드 괄호안에 그것을 넣으면 된다.

()괄호안에는 CSS, JavaScript구문도 넣을수 있다.

실례:

```
<script language="JavaScript">
<!--
document.write(window.prompt("안녕하십니까", "임의의 글자를
넣어보십시오."))
-->
</script>
```

괄호()안에 JavaScript구문을 넣을 때에는 인용부호를 넣지 않아도 되지만 꼬리표나 일반적인 본문을 넣을 때에는 인용부호를 넣어야 한다.

이번에는 alert()에 넣어보자.

실례:

```
<script language="JavaScript">
<!--
window.alert(window.prompt("안녕하십니까","임의의 글자를
넣어보십시오."))
-->
</script>
```

입력한 내용이 alert의 경고창에 나오게 된다.

실례:

```
<script language="JavaScript">
<!--
a=prompt("1+1은?","여기에 답을")
b=confirm("정확합니까?")
if(b==true){
if(a==2){alert("옳음")} else{alert("틀렸음")}
}
else{
alert("확신을 가지십시오")
}
-->
</script>
```

⑥ setTimeout()메쏘드

setTimeout()메쏘드는 window객체의 메쏘드로서 일정한 시간이 지나면 자동적으로 다시 적재해주는 기능을 수행한다. 문서자체를 다시 적재할수도 있으며 자그마한 그림만 다시 적재할수도 있다. 물론 그 메쏘드는 함수나 기타 JavaScript구문으로 조종한다.

형식은 다음과 같다.

```
setTimeout("JavaScript구문",시간)
```

JavaScript구문에는 다시 적재할 목표를 넣으면 된다. 그리고 그 뒤의 시간은 다시 적재를 하는 시간인데 그 단위는 1/1000s로 정해져있다. 따라서 300으로 주면 0.3s에 한번씩 다시 적재한다.

실례:

```
<script language="JavaScript">
<!--
window.setTimeout('document.write("안녕하십니까")',1000)
-->
</script>
```

1s가 지나면 《안녕하십니까》가 출력되도록 한 구문이다.

setTimeout()의 인용부호와 write()의 인용부호를 한 구문에 같이 쓰므로 구별해주어야 한다.

실례:

```
<script language="JavaScript">
```

```
<!--
window.setTimeout('window.close()',2000)
-->
</script>
```

이것을 실행하면 2s후에 자동으로 닫긴다.

실례:

```
<script language="javascript">
function abcd(){
today = new Date()
document.title = "지금 시각" + today.getHours() + "시"+
today.getMinutes()+ "분" + today.getSeconds()+"초 이다."
setTimeout("abcd()", 1000)
}
</script>
<body onload="abcd()">
```

이 실례를 실행하면 창문제목띠에 현재시간이 나온다.

이 코드의 내용을 구체적으로 보기로 하자.

우선 <body>꼬리표에서 강제로 abcd()라는 함수를 호출하면 호출된 함수는 document.title에 의해 제목띠에 문자를 표시한다. 여기서 date객체에 의한 getHours(), getMinutes(), today.getSeconds()의 값을 얻어서 창문의 제목띠에 현재시각을 나타내도록 하고있다. 또한 setTimeout("abcd()", 1000)에 의해서 1s에 한번씩 abcd()함수를 계속 다시 적재하게 함으로써 현재시각이 1s에 한번씩 새로 표시된다. 따라서 결과적으로 제목띠가 시계처럼 계속 변하게 되는것이다.

document.title이 아니라 window.status로 한다면 상태띠에 현재 시간이 나타난다.

실례:

```
<script language="JavaScript">
<!--
var pos = 0
var a = 0
var message = new Array()
message[0] = "HTML TUTOR에 온것을 환영합니다."
message[1] = "JavaScript가 어렵습니까?"
function abcd()
```

```
{
a = a + 1
document.title = message[pos].substring(0, a)
if(a == message[pos].length + 5)
{
pos = pos + 1
a = 0
}
if(pos > 1)
{
pos = 0
}
setTimeout("abcd()", 200)
}
-->
</script>

<body onLoad="abcd()">
```

3.2.4. date 객체

date()객체는 문자그대로 날자와 시간에 관한 정보를 포함하는 객체이다. date객체는 콤퓨터에 이미 내장되여있는 시간정보를 알아내는데 자주 사용된다.

여기서 중요한것은 여러가지 객체는 document.write()나 window.open()와 같은 형식으로 객체의 바로 뒤에 메쏘드나 속성이 붙었지만 date객체는 date.getHours()형식으로 뒤에 메쏘드나 속성이 붙지 않는다는것이다. 따라서 date객체를 대신할 변수가 필요하게 되는데 우의 실례에서 본 today변수가 바로 그러한 변수이다. 이것을 객체의 선언이라고 한다.

이렇게 하여 today라는 변수가 앞으로는 date객체의 역할을 대신하게 된다. 여기서는 메쏘드를 붙일수 있다. 즉 today.getHours()라고 써주면 된다. 물론 메쏘드의 선언도 할수 있다.

var hour = today.getHours()은 hour라는 변수가 getHour()메쏘드를 대신한다는 뜻이다. 물론 new연산자는 제외한다.

현재년도를 알아내는 getYear()메쏘드도 있다. 날자와 관련된 정보는 열람기에 표시해서 알아내는 용도로 많이 쓰이게 된다. 따라서 문서의 표현과 관련된 document.write()와 함께 사용된다.

실례:

```
<script language="JavaScript">
<!--
var today = new date()
document.write(today.getYear())
-->
</script>
```

이 실례에서는 date객체에 직접 메쏘드를 붙일수 없으므로 today라는 변수로 대신하고 여기에 getYear()메쏘드를 붙여 today.getYear(), 즉 현재년도를 나타내고있다.

그러면 메쏘드들에 대하여 보기로 하자. 메쏘드들에서는 대소문자를 구별하여야 한다.

① getYear(): 현재년도를 얻기
② getMonth(): 현재달을 얻기(값이 0이면 1월, 1은 2월, ...11은 12월)
③ getDate(): 현재날자를 얻기
④ getDay(): 현재요일를 얻기(값이 0이면 일요일, 1은 월요일, ...6은 토요일)
⑤ getHours(): 현재시간의 시만을 얻기
⑥ getMinutes(): 현재시간의 분만을 얻기
⑦ getSeconds(): 현재시간의 초만을 얻기

실례:

```
<script language="JavaScript">
<!--
var today = new date()
document.write(today.getYear() + "<br>")
document.write(today.getMonth() + 1 + "<br>")
document.write(today.getDate() + "<br>")
document.write(today.getDay() + "<br>")
document.write(today.getHours() + "<br>")
document.write(today.getMinutes() + "<br>")
document.write(today.getSeconds() + "<br>")
-->
</script>
```

차례대로 년, 월, 일, 요일, 시, 분, 초를 얻는다. getMonth()뒤에 하나를 더한것은 0의 값이 1월이고 1의 값이 2월이므로 메쏘드값과 달을 맞춰주기 위해서이다.

이번에는 우의 메쏘드들을 모두 변수로 바꾸어보자

실례:

```
<script language="JavaScript">
<!--
var today = new date()
var a = today.getYear()
var b = today.getMonth()
var c = today.getDate()
var d = today.getDay()
var e = today.getHours()
var f = today.getMinutes()
var g = today.getSeconds()
document.write(a + "<br>")
document.write(b + 1 + "<br>")
document.write(c + "<br>")
document.write(d + "<br>")
document.write(e + "<br>")
document.write(f + "<br>")
document.write(g + "<br>")
-->
</script>
```

선언의 개념을 알고 이와 같이 메쏘드나 객체를 변수로 지정해주면 긴 구문을 작성할 때 편리하다.

실례:

```
<script language="JavaScript">
<!--
var today = new date()
var a = today.getDay()
if (a == 0) {var b = "일요일"}
if (a == 1) {var b = "월요일"}
if (a == 2) {var b = "화요일"}
if (a == 3) {var b = "수요일"}
if (a == 4) {var b = "목요일"}
if (a == 5) {var b = "금요일"}
```

```
if (a == 6) {var b = "토요일"}
document.write("오늘은" + b)
-->
</script>
```

이렇게 많은 객체와 메쏘드, 선언, 함수 등을 섞어서 어려운 효과도 낼수 있게 하는것이 바로 JavaScript이다.

제3절. JavaScript자료형과 연산자

3.3.1. 자료형

JavaScript뿐만아니라 많은 프로그람언어는 문자렬이나 수값 등의 값을 다른 형식으로 표시할수 있다.

- 문자렬형

JavaScript에서는 두점인용부호(“) 혹은 인용부호(‘)안에 있는 값을 문자렬로 취한다.

실례:

“123”   “konnichiha”   ‘문자렬’

- 수값형

수값에는 정수형과 류동소수점형이 있다.

- 정수형

정수로는 8진수, 10진수, 16진수를 사용할수 있다.

실례:

0514, 156, 0x11

- 류동소수점형

류동소수점은 소수점을 점(.)으로서 표시한 10진수, 또는 지수를 말한다.

- 론리값형

론리값이란 설정한 값이나 식이 참인가 거짓인가에 따라 표시되는 값이다.

참은 값이나 조건식이 옳거나 혹은 조건에 맞을 때를 의미하며 거짓은 식이나 조건식이 옳지 않거나 혹은 조건에 맞지 않을 때를 말한다. 참은 true, 거짓은 false라는 값으로서 표시된다.

론리값을 리용하여 어떤 식이 true일 때에는 처리 A를 진행하고 false일 때에는 처리 B를 진행한다.

-null값형

null값은 값이 아무것도 설정되여있지 않을 때나 미정일 때를 표시한다.

실례:

Window객체의 onError사건에 null값을 설정하자.

이 스크립트를 HTML문서의 제일 처음에 서술하면 스크립트에 문제가 있어 오유가 생겼다고해도 열람기에는 오유창문이 표시되지 않게 된다.

즉

```
<script language=" JavaScript1.2" >
<!--
window.onError=null
//-- >
</script>
```

3.3.2. 연산자

프로그람작성에서는 여러가지 값을 더하거나 덜거나 일정한 조건에서 계산하는 처리를 진행할수 있다. 또한 비교나 선택과 같은 처리도 진행한다. 이와 같은 식표현에 리용하기 위한 기호를 연산자라고 말한다.

JavaScript에서 쓰이고있는 연산자들은 다음과 같다.

-산수연산자

표 3-2. 산수연산자

기호	역할
=	변수에 값을 대입
+	더하기
-	덜기 혹은 부의 값을 표시
*	곱하기
/	나누기
%	나누기에서 나머지가 나왔을 때에는 소수점아래를 자르고 값을 현시
++	값을 하나 증가한다.
--	값을 하나 감소한다.

실례:

```
<script language= "JavaScript" >
<!- -
```

```
    a = 1+2 ;
    document.write(a);
  //-- >
  </script>
```

- 비교연산자

량쪽의 값을 비교하기 위한 연산자를 비교연산자라고 한다.

량쪽의 값을 비교하고 참인 경우 true, 거짓인 경우 false의 값이 얻어진다.

표 3-3. 비교연산자

문법	연산결과가 true인 경우의 의미
x==y	x는 y와 같다.
x!=y	x는 y와 같지 않다.
x<y	x는 y보다 작다.
x<=y	x는 y보다 작거나 같다.
x>y	x는 y보다 크다.
x>=y	x는 y보다 크거나 같다.

- 론리연산자

량쪽의 값을 론리적으로 비교하기 위한 연산자를 론리연산자라고 한다. 례를 들면 x&&y라는 식은 x와 y의 값이 참일 때에만 true로 되고 그 외에는 거짓(false)으로 된다.

식을 x||y라고 쓴 경우는 x와 y값이 둘다 참인 경우 또는 하나가 참인 경우 true로 된다.

식 x!y라고 씌여진 식은 x와 y값이 거짓(false)일 때에 true이다.

표 3-4. 론리연산자

문 법	의 미
x&&y	x 그리고 y (AND)
x||y	x 혹은 y (OR)
x!y	x는 y가 아니다.(NOT)

-문자렬연산자

"+" 기호를 문자렬과 문자렬사이에 사용하면 그 문자렬은 하나로 련결된다. 문자렬을 련결하는 이 연산자를 문자렬연산자라고 한다.

표 3-5. 문자렬연산자

문 법	의 미
"문자렬A" + "문자렬B"	문자렬A와 문자렬B를 련결하다.
a+= "문자렬B"	a의 뒤에 문자렬B를 추가하다.

-비트연산자

콤퓨터안에서는 모두 0과 1의 비트단위로 처리되고있다.

비트연산자는 비트단위로 처리를 진행하기 위한 연산자이다.

표 3-6. 비트연산자

기 호	역 할
~	비트의 반전
&	비트의 론리적(AND)
|	비트의 론리합(OR)
^	비트의 배타적합(XOR)
<<	비트의 왼쪽밀기
>>	비트의 오른쪽밀기
>>>	비트의 론리오른쪽밀기
<<=	비트의 왼쪽밀기의 대입
>>=	비트의 오른쪽밀기의 대입
>>>=	론리오른쪽밀기의 대입

- typeof()연산자

typeof()연산자는 수값, 문자렬, 변수, 오브젝트 등의 형을 얻어내는 연산자이다. JavaScript1.1에서부터 추가되였다.

실례:

```
<script language= "JavaScript" >
<!- -
 var suuzi=123
 var mozi= "오늘은"
 var kye=null
 var today=new date()
```

```
var kan=blur
document.write(typeof(suuzi)+ “<br>”)
document.write(typeof(mozi)+ “<br>”)
document.write(typeof(kye)+ “<br>”)
document.write(typeof(today)+ “<br>”)
document.write(typeof(kan)+ “<br>”)
document.write(typeof(muimi))
//-- >
</script>
```

- void()연산자

void()연산자는 되돌림값이 없는 식이나 함수를 처리하는 연산자이다. JavaScript1.1에 추가되였다.

실례:

<a href= “javascript:void(kansuu())” >이 문자를 누르면 kansuu()가 발생한다.</A>

연산자의 우선순위는 다음과 같다. 여기서 행에 놓이는 연산자들은 같은 우선순위를 가진다.

```
() ,[]
! ,~ ,++, --, typeof, void
*, /, %
+, -
<<, >>, >>>
<<= ,>>=
== ,!=
&
^
|
&&
||
?:
 =, +=, -=, *=, /=, %=, <<=, >>=, >>>=, &=, ^=, |=
```

제4절. JavaScript명령문들

3.4.1. 조건분기명령문

if문은 어떤 조건에 관한 처리를 진행할 때 사용한다.

기본형식은 다음과 같다.

```
if(조건식){처리1}
        else {처리2}
```

if문은 조건식이 참(true)일 때에 《처리 1》을 진행하고 그 외에는 《처리 2》를 진행한다. 조건이 여러개인 경우에는 if(조건식){처리1}를 여러개 결합하여 처리한다. 그리고 괄호 {}은 생략할수도 있다.

만일 조건식에 맞지 않을 때 아무런 처리도 하지 않도록 하자면 else의 처리를 생략하면 된다.

실례:

```
<script language= "JavaScript" >
<!- -
var now=new date();
var AM_PM=now.getHours();
if (AM_PM<12){document.write( "오전" )}
else{document.write( "오후" )}
//-- >
</script>
```

if문은 다음과 같이 여러개 동시에 쓸수 있다.

```
if (조건식 A) {if (조건식 B) {처리 1} }
   else {처리 2}
```

3.4.2. 선택명령문

어떤 조건이 있고 그것이 참(true)일 때와 거짓(false)일 때에 서로 다른 처리를 진행하도록 할수 있다.

그 형식은 다음과 같다.

```
조건식 ? x:y
```

실례:

```
<script language= "JavaScript" >
<!- -
 var now=new date();
 var AM_PM=now.getHours();
     document.write(AM_PM<12 ? "오전" :" 오후");
//-- >
</script>
```

-switch문

switch문은 처리에 《표식》을 설정하고 《값》과 매 《표식》을 비교하여 참(true)으로 되면 그 《표식》의 처리를 실행한다. 만약 참(true)으로 되는 《표식》이 없는 경우는 《default:》의 처리를 실행한다.

형식은 다음과 같다.

```
switch (값) {
      case 표식A:
            처리;
            break;
      case 표식B:
            처리;
            break;
              .
              .
              .
      default:
            처리;
      }
```

3.4.3. 순환명령문

- for명령문

이 명령문은 증감식을 설정하고 그것에 따라서 어떤 조건을 만족할 때까지 같은 처리를 반복진행하도록 하는 경우에 사용한다.

형식은 다음과 같다.

```
for (초기값; 조건식; 증감식) {처리}
```

for문은 증감식에 따라서 값이 변경되며 조건식이 참(true)일동안 처리를 반복진행한다.

for문이나 while문, do while문 등은 일정한 조건이 만족될 때까지 반복처리를 진행하도록 하는 명령문이므로 증감식과 조건식의 정합성에 주의를 돌려야 한다.

례를 들어 《for(i=1; i<=10; i--)》라고 하면 조건식은 참으로 될수 없고 반복조작처리를 끝낼수 없으며 그 결과 체계가 파괴되는 현상이 나타날수 있다.

- while명령문

이 명령문은 어떤 조건식을 만족할 때까지 같은 처리를 반복하는 경우에 사용한다.

형식은 다음과 같다.

```
While (조건식) {처리}
```

이 경우 조건식이 참(true)인동안 처리를 반복진행한다.

실례:

```
<script language= "JavaScript" >
<!--
 var i=1
 while(i<=10){
document.write( "이 문장을 10번 씁니다:" +i+ "번째<br>" );
i++;
//-- >
</script>
```

-do while명령문

이 명령문은 while명령문과 거의 류사하며 다른점은 while명령문처럼 조건식의 값을 먼저 판단하는것이 아니라 무조건 먼저 처리부분을 집행한 다음 조건식의 값을 판단한다. 즉 처리부분이 적어도 한번은 집행된다.

형식은 다음과 같다.

```
do 처리
        while (조건식);
```

실례:

```
<script language= "JavaScript1.2" >
<!--
var i=1;
do {
```

```
document.write( "이 문장을 10번 씁니다.:" +i+ "번째<br>" );
 i++;
} while (i <=10);
//-- >
</script>
```

-break명령문

이 명령문은 for문이나 while문에서 사용한다. break명령문을 반복조작(순환)처리를 진행하는 부분에 넣고 어떤 조건이 참(true)으로 되면 순환에서 탈퇴하는 처리를 진행할수 있다.

실 례:

```
<script language= "JavaScript" >
<!--
   for (i=1; i<=10; i++){
      document.write( "이 문장을 10번 씁니다:" +i+ "번째" );
      if (i==5)
         break;
   }
document.write( "<br>에서 break가 있으므로 순환을
탈퇴합니다.<br>" );
//-- >
</script>
```

-continue명령문

이 명령문은 for문이나 while문 등에서 반복조작(순환)처리를 진행하는 도중에 설정한 조건이 참(true)으로 되였을 때에만 continue아래의 처리를 뛰여넘어서 진행한다. continue는 처리를 중단시키지는 못하므로 련속하여 반복조작처리가 계속된다.

실례:

```
<script language= "JavaScript" >
<!--
 for (i=1; i<=10; i++){
if (i==5)
 continue;
document.write( "이 문장을 10번 씁니다:" +i+ "번째에서도 …<br>" );
}
document.write( "<br>continue가 있으므로 5번째가 없습니다.<br>" );
```

```
//-- >
</script>
```

- 표식을 붙여 순환고리나 조건분기명령문에서 탈퇴하기

먼저 표식명을 입력하고 break나 continue명령문에 의해 if문이나 순환문으로부터 빠져나올 때 지적한 표식명을 지정하여 명시적으로 탈퇴하는 장소를 지정할수 있다.

형식은 다음과 같다.

```
표식명:
식
break 표식명;

표식명:
식
continue 표식명;
```

- return명령문

이 명령문은 명령문안에서 값을 돌려줄 때에 사용한다.

형식은 다음과 같다.

```
return 값
```

실례:

이 문장우에 마우스를 가져가면

마우스를 해당 위치에 가져다 놓을 때 onMouseOver에 따라서 사건이 취급되며 상태띠에 문자를 표시하는 처리에 대하여 참(true)의 값을 주므로써 창문의 상태띠에 문자를 표시한다.

3.4.4. 객체의 생략문(with)과 참조문(this)

- 생략문 with

with문은 객체를 생략할 때에 사용한다.

형식은 다음과 같다.

```
with(객체) {처리}
```

실 례:

```
<script language= "JavaScript" >
<!--
```

```
with(document) {
write( “이 문장은 document를 <br>” );
write( “생략하고있다.<br>” );
}
//-- >
</script>
```

원래 《document.write()》라고 서술하여야 하겠는데 실례에서는 with를 사용함으로써 《document.》부분을 생략하고있다.

- 참조문(this)

this는 현재객체에 대한 인용을 말한다. 이것을 통해 객체에 접근할수 있다.

실 례:

```
<html>
<head>
<title></title>
<script language= “JavaScript” >
<!- -
function Namae(n){alert( “입력된 이름은” +n+ “이다.!!” )}
//-- >
</script>
</head>
<body>
이름을 입력하십시오.<br>
<form name= “name” >
<input type= “text” name= “Namae” onBlur= “Namae(this.value)” value= “” >
</form>
</body>
</html>
```

제5절. 응용실례(달력만들기)

```
<html>
<head>
<script language= “JavaScript” >
<! - -
//시간을 오전/오후로 열람기에 보여줄수 있게 해준다.
function get_Time()
{
    var now = new date()
    var hour = now.getHours()
    var minute = now.getMinute()
    var ampm
    now = null
if(hour>= 12){
        hour -= 12
        ampm =  “오후”
    } else
        ampm =  “오전”
    hour = (hour == 0) ? 12 : hour
 if(minute < 10)
        minute =  “0”  + minute
    return ampm + hour +  “ :  “ + minute
}
// 년과 달을 받아서 마지막 날자를 알아낸다.
function get_Day(year, month)
{
    var Last_Mon = new Array(31,29,31,30,31,30,31,31,30,31,30,31)
    var Mon2
  if(year % 4 == 0)
        Mon2 = true
    else
        Mon2 = false
    Last_Mon[1] = (Mon2) ? 29 : 28
```

```
        return Last_Mon[month]
    }
    // <table>을 리용하여 달력을 만들어준다.
    Function drawCal(firstDay, lastDate, date, year, monthName)
    {
         var text = “”
         text += “<center>”
         text += “<table>”
         text += “<th colspan=7 bfcolor=#f0f0f0>”
         text += “<font color=midnightblue size=+3 >”
    // 년과 월을 출력합니다.
         text += year + “년 “ + (monthName + 1) + “월”
         text += “</font>”
         text += “</th>”
        var openCol = “<td bgfcolor=#ffefff width=45 height=40>”
         openCol += “<font color = darkblue>”
        var closeCol = “</font></td>”
         text += “<tr align = center valign = center>”
        var weekDay=new Array
                     (“일”,“월”,“화”,“수”,“목”,“금”,“토”)
                    // 달력의 일, 월, 화, 수, 목, 금, 토요일을 출력한다.
        for (var dayNum = 0 ; dayNum <= 6 ; dayNum++)
            text += openCol + weekDay[dayNum] + closeCol
            text += “</tr>”
        var digit = 1
        var curCell = 1
        //달력 표를 만들어줍니다.
        for(var row = 1 ; row <=
                  Math.ceil((lastDate + firstDay    1) / 7) ; ++row)
        {
            text += “<tr align=right valign=top bgcolor=#ffefee>”
            for (var col = 1 ; col <= 7 ; col++)
                if(digit > lastDate) break
                if(curCell < firstDay)
```

```
            {
                 text +=  "<td>   </td>"
                 curCell ++
            }
            else
             {
                  if(digit == date)
                     {
                          text +=  "<td height = 40>"
                          text +=  "<font color = Red>"
                          text += digit
                          text +=  "</font><br>"
                          text +=  "<font color=purple size=2>"
                          text += "<center>" + get_Time() + "</center>"
                          text +=  "</font>"
                          text +=  "</td>"
                      }
                      else
                          text +=  "<td height=40>"  + digit +  "</td>"
                       text +=  "<td height=40>"  + digit +  "</td>"
                      digit++
                    }
               }
              text +=  "</tr>"
         }
      text +=  "</table>
      text +=  "</center>
      return text
     }
//- ->
</script>
</head>
<body>
<script language= "javascript" >>
```

```
<!- -
    var now = new Date()
    var year = now.getYear()
    var month = now.getMonth()
    var date = now.getDate()
    var my_text
     now = null
   var firstDayInstance = new Date(year, month, 1)
   var firstDay = firstDayInstance.getDay() + 1
    firstDayInstance = null
  var days = get_Day(year, month) // 달의 마지막 일을 구한다.
    my_text = drawCal(firstDay, days, date, year, month)
  //최종적으로 만들어진 HTML문서를 열람기에 출력한다.
    document.write(my_text)
 //- ->
</script>
</body>
</html>
```

제 4 장. ASP 언어

제1절. ASP언어의 기초

4.1.1. ASP에 대한 일반개념

ASP(Active Server Page)란 VBScript를 기반으로 하는 봉사기측스크립트를 실현하고 이 Server Script로 자료기지나 다른 외부 부분품들을 불러들여 강력한 홈페지를 구축하는 웨브봉사기기술의 한가지로서 동적인 웨브페지를 작성할수 있다.

이전의 스크립트들은 모두 의뢰기에서 실행되였다. 즉 봉사기는 단순히 의뢰기에 스크립트원천프로그람을 전송해주며 이것을 전송받은 의뢰기는 열람기에 스크립트를 분석하여 처리하여주었다. 그런데 열람기들은 모든 스크립트들을 다 리해하지는 못한다. 다시 말하여 열람기마다 리해할수 있는 스크립트가 다르다. 실례로 Java Script는 Netscape나 Internet Explorer가 다 처리할수 있지만 VBScript는 Internet Explorer만이 처리할수 있다.

또한 의뢰기측스크립트파일들은 모두 공개되여있다.

지정된 주소에 있는 일반 HTML문서들은 단순히 봉사기에 의해 의뢰기측에 전송되여 현시된다.(그림 4-1)
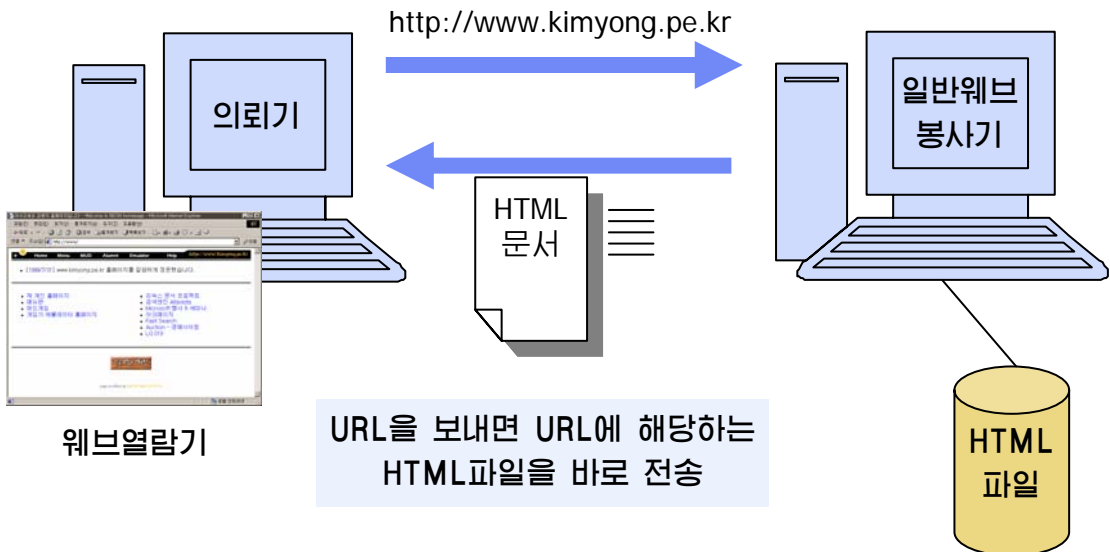
그림 4-1. 일반 HTML문서봉사형식

ASP는 Microsoft회사에서 나온 웨브봉사기인 IIS(Internet Information Server:인터네트정보봉사기)에 확장모듈로 내장되여있다. 요청이 들어오면 이 봉사기에서 처리하여 그 결과를 HTML문서로 변환하여 열람기에 전송한다.

그 과정을 그림 4-2에 보여주었다.
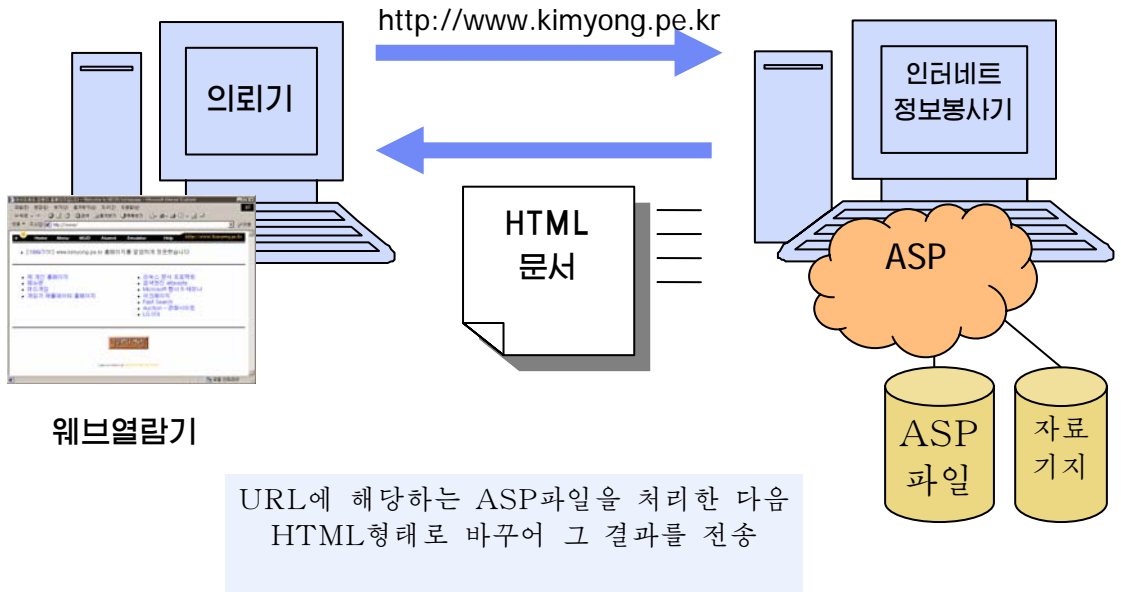
그림 4-2. ASP문서봉사형식

4.1.2. ASP의 특징

첫째로, ASP는 Microsoft회사에서 만든 스크립트엔진으로서 항상 같은 내용만을 보여주는 일반 HTML문서와는 달리 사용자의 요구에 따라 봉사해줄수 있는 동적인 웨브페지를 구축할수 있다.

둘째로, ASP는 NT에 설치되는 웨브봉사기인 IIS(Internet Information Server)에서 동작한다. 만일 NT가 설치되지 않은 콤퓨터인 경우에는 PWS(Personal Web Server)를 더 설치하여야 한다.

셋째로, VBScript를 기본으로 만들어진 강력한 봉사기측스크립트로서 사용자에게는 일반 HTML코드내용만이 보여지고 봉사기측스크립트부분은 볼수 없게 되여 보안측면에서 아주 유리하다.

넷째로, 기본적으로 제공되는 ADO부분품을 리용하여 단순히 메쏘드를 호출하는 방식으로 자료기지에 접근하므로 자료기지를 사용하기 쉽다.

제2절. ASP언어의 문법

4.2.1. 변수와 상수, 배렬, 함수

변수는 어떤 값을 담아두는 그릇처럼 자주 변하는 프로그람정보를 저장할수 있는것을 말한다.

VBScript에서는 기본적으로 Variant형변수를 사용한다. 이 Variant형변수는 사용방법에 따라 여러가지 자료형의 정보를 저장할수 있다. 즉 Variant형변수는 수값을 저장하면 수값형변수로 되고 문자렬자료를 저장하면 문자렬형변수로 된다.

여러가지 류형의 변수가 있다.(표 4-2)

표 4-1. 변수종류

종 류	의 미
integer	-32768~32768사이의 옹근수
double	배정확도실수형
single	단정확도실수형
byte	0~255사이의 정수
currency	화폐형
long	-2147483648~2147483648사이의 옹근수형
string	문자렬형
boolen	true나 false만을 가지는 론리형
dateTime	날자시간형
empty	변수값이 주어지지 않은 형
null	자료값이 주어지지 않은 형

변수선언은 Dim을 리용하는 방법과 암시적으로 선언을 하지 않고 직접 변수를 리용할수도 있으며 Option Explicite명령을 리용할수도 있다.

Dim으로 변수선언을 진행하면 프로그람의 유지 및 보수, 오유도 쉽게 찾을수 있다.

실례:

```
Dim TeamType
TeamType = “Nolja”
```

Option Explicite명령을 리용하여 변수선언을 하는 경우 ASP스크립트의 첫 문장에 표기하여야 하며 철저히 변수를 사용하기전에는 Dim으로 변수선언을 하여야 한다. 이

방법을 리용하면 코드실행속도를 저하시키는 요소를 추적하기 쉽다.

실례:

```
<% Option Explicit
Dim intA, intB, intT,intC
intA = 20
intB = 30
intC = 30
intT = intA + intB - intC
Response.Write intT
%>
```

배렬을 리용하면 한개의 변수에 여러개의 관련자료들을 저장할수 있으며 매개 값들은 첨수번호에 의해 구분된다. 배렬 역시 Dim으로 선언하며 항목개수를 먼저 지정했으나 이 값들을 바꿀 필요가 있을 때 ReDim을 리용하여 재선언한다. 자료를 보존하고 있는채로 배렬의 크기를 변화시키려면 ReDim과 함께 Preserve를 함께 사용하면 된다. 배렬은 다차원배렬도 지원하는데 현실적으로 2차원, 3차원배렬이 가능하다.

실례:

```
<html>
<head>
<title>배렬 실례</title>
</head>
 <body>
 <P>여기 우리 형제들의 이름이 있습니다.</p>
 <%
 Dim strBrother()
 Redim strBrother(4)
 strBrother(0) = "영철"
 strBrother(1) = "영남"
 strBrother(2) = "영호"
 strBrother(3) = "영식"
 Response.Write "<p><b>"
 For intNum = 0 to 3
 Response.Write strBrother(intNum) & "..."
 Next
```

```
Response.Write "</b></p> "
Response.Write "<p>막내이름이 빠졌군!</p> "
Redim Preserve strBrother(5)
strBrother(4) = "영선"
Response.Write "<p><b>"
For intNum = 0 to 4
Response.Write strBrother(intNum) & "..."
Next
%>
</b></p>
</body>
</html>
```

상수는 프로그람에서 값을 할당받으면 변하지 않는 수로서 const를 사용하여 상수를 정의하며 이름은 대체로 대문자로 쓴다.

실례:

```
Const COUNTNUM = 37
```

문자렬을 처리하는 함수들은 Lcase()와 Ucase()들이며 문자렬길이를 얻는 함수 Len(), 문자렬의 몇개 문자 추출함수인 Left(string, 문자개수)와 Right(string, 문자개수), 문자렬의 중간부분추출함수인 Mid(string, 문자위치, 추출문자개수), 특수한 단어찾기함수인 InStr(string, 찾을 단어), InStraRev(string, 찾을 본문) 등 문자렬관련함수들이 여러가지가 있다.

함수의 표시형식은 다음과 같다.

```
Function 함수명 (인수값)
명령문
함수명=일정한 식
End Function
```

인수값이 없으면 빈괄호, 인수값이 여러개이면 반점으로 구분한다.

함수이름 다음의 괄호안에 귀환값을 입력함으로써 함수호출변수에 함수의 귀환값을 지정할수도 있다.

실례:

```
<html>
<head>
<title>함수</title>
</head>
<body>
<%
Function UsedDate(varCKout)
UsedDate = varCKout + 15
end function
%>
<h2>보름 후의 날자를 알려준다.</h2>
<%
CKout = date()
varUsing = UsedDate(CKout)
Response.Write "오늘은 " & CKout
Response.Write "이고"
Response.Write "<br> 보름후의 날자는 " & varUsing
Response.Write " 이다. "
%>
</body>
</html>
```

4.2.2. 연산자

대입연산자는 변수를 선언할 때 변수에 값을 할당하는 연산자이다. 대입연산자는 《=》이다.

형식은 다음과 같다.

```
변수이름 = 변수값
```

실례:

```
<%Dim Num1,strNum1,strNum2
Num1 = 2
strNum1 = Num1
Response.Write strNum1 & "<br>"
Num1 = Num1 + 1
strNum2 = Num1
```

```
Response.Write strNum2
%>
```

비교연산자는 비교연산을 진행하는 연산자로서 비교연산결과는 량변의 비교식의 결과에 따라 true이거나 false이며 또는 null일수도 있다.

null값은 비교식중 하나가 null인 경우이다.

표 4-2. 비교연산자

연산자	의미
=	같다.
<>	같지 않다.
<	보다 작다.
<=	보다 작거나 같다.
>	보다 크다.
>=	보다 크거나 같다.

실례:

```
<%
 Dim Num1,Num2,Num3,Num4,strNum
 Num1 = 1
 Num2 = 2
 Num3 = 3
 Num4 = 4
 If Num1 <> Num2 then
     Response.Write Num1 & "과" & Num2 & "는 다름<br>"
 end if
 If Num3 <= Num4 then
     Response.Write Num3 & "은" & Num4 & "보다 작거나 같음<br>"
 end if
 strNum = Num3 + 1
 If strNum = Num4 then
     Response.Write strNum & "과" & Num4 & "는 같음<br>"
 end if
 %>
```

수식연산자에는 다음과 같은것들이 있다.(표 4-3)

표 4-3. 수식연산자

연산자	의 미
+	더하기
-	덜기
*	곱하기
/	나누기
^	제곱
-	부수
\	정수나누기
MOD	나머지연산

실례:

```
<%
 Dim Num1,Num2,Num3,Num4,NumT
 Num1 = 1
 Num2 = 2
 Num3 = 3
 Num4 = 4
 NumT = Num1 + Num2
 Response.Write Num1 & "+" & Num2 & "=" & NumT & "<br>"
 NumT = Num3 - Num1
 Response.Write Num3 & "-" & Num1 & "=" & NumT & "<br>"
 NumT = Num2 * Num4
 Response.Write Num2 & "x" & Num4 & "=" & NumT & "<br>"
 NumT = Num4 / Num2
 Response.Write Num4 & "/" & Num2 & "=" & NumT & "<br>"
 %>
```

론리연산자는 한개 또는 그 이상의 수식에서 값의 참과 거짓을 판별하는 연산자로서 여기에는 AND, OR, NOT, EQV, IMP, XOR가 있다. 문자렬들을 련결하는 연산은 《&》를 리용하여 한다.

4.2.3. 조종문

-If문

형식은 다음과 같다.

```
① If 조건식 then 명령문
 End if
② If 조건식 then
 명령문1
 명령문2
End if
③ If 조건식 then
 명령문1
 명령문2
   Else
 명령문3
 명령문4
 End if
④ If 조건식 then
 명령문1
 명령문2
   Else if
 명령문3
 명령문4
   Else if
 명령문5
 명령문6
 End if
```

여기서 주의해야 할것은 《End if》를 《Endif》와 같이 붙여쓰면 오유가 발생한다는것이다.

-선택문

Select Case문은 여러가지 종류중에서 선택을 할 필요가 있을 때 사용한다. End Select로 이 문을 끝낸다.

Case Else로 특별한 조건을 작성할수도 있다.

실례:

```
<%
Select Case dropdown
      Case "1"
            Response.Write "Desk"
      Case "2"
            Response.Write "Chair"
      Case "3"
            Response.Write "Book"
End Select
%>
```

순환명령문으로서 Do While…Loop문, Do…Until, For…Next, For Each…Next 등이 있다. 여기서 For Each…Next는 배렬이나 묶음안의 각 항목들에 대해서 순환할 필요가 있을 때 사용한다.

제3절. ASP의 내장객체들

4.3.1. Request 객체

Request객체는 의뢰기가 페지를 요청하거나 홈을 보낼 때 묶음들에 의뢰기가 제공하는 모든 정보를 저장하는 객체이다.

Request객체가 가지고있는 기능들은 표 4-4와 같다.

표 4-4. Request의 기능

종 류		설 명
묶음	ClientCertificate	HTTP요청으로 봉사기에 전달된 의뢰기인증서에 저장된 값들의 모임
	Cookie	HTTP머리부와 함께 전송된 쿠키값
	Form	Post방법으로 전송되여온 HTTP요청의 요소값
	QueryString	Get방법으로 전송되여온 값과 URL의 HTTP요청문자렬의 변수값
	ServerVariables	봉사기에서 참조되는 HTTP머리부들 및 환경변수값
속성	TotalBytes	의뢰기가 보내는 HTTP요청의 총 바이트수
메쏘드	BinaryRead	홈에서 Post방식으로 봉사기에 보내는 자료를 불러들이는데 사용

Request객체에서 묶음의 원소들을 찾는 순서는 그림 4-4와 같다.

그림 4-4. Request에서 묶음원소를 찾는 순서

Request객체는 HTML에서 Post, Get, Cookie 또는 사용자인증방식에 의해 전달되는 HTTP의 요청에 대한 정보를 획득하는데 사용된다.

Request객체를 사용하여 의뢰기의 정보를 봉사기로 전달하자면 홈의 두가지 전송방식을 잘 알아야 한다. 두 방식인 Post방식과 Get방식은 의뢰기의 정보를 봉사기로 전송하는데서 현저한 차이를 가지고있다.

Get방식은 사용자가 입력한 자료를 URL에 붙여서 전송하는데 스크립트에서는 《QueryString(“name”)》으로 작성되여 사용자의 입력자료를 처리한다. 이 방식을 리용하면 URL부분으로 정보가 전달되므로 중간에서 류실될수도 있고 변경될 가능성이 있으므로 중요한 자료들을 전송하는데는 적합치 못하다.

여기서 QueryString묶음은 봉사기에 《이름=값》의 형태로 전달되는 정보로서 URL에 ?표시와 함께 추가된다. 전달되는 정보들이 한개 이상일 때에는 &기호로 분류하며 서로 다른 값이 계속 같은 이름에 할당되여도 원래 값이 상실되지 않고 계속 추가된다. 이런 경우에는 이름과 값들의 쌍개수를 알려주는 Count속성으로 표현한다.

실례:

<%=Request.QueryString(“name”).Count %>

Post방식은 HTTP머리부에 의뢰기의 정보를 넣어서 전달하므로 보안상 자료의 류실을 막을수 있다. 스크립트에서는 《Form(“name”)》으로 작성되여 사용자의 입력자료를 처리한다.

일반적으로 홈을 리용하여 자료를 입력받고 전송하는 경우에는 거의 Post방식을 리용하고 하이퍼련결로 정보를 보내는 경우에는 Form묶음으로는 그 값들을 받아오지 못하기때문에 QueryString이 쓰인다.

Request.QueryString나 Request.Form을 사용하는 경우 그냥 Request라고만 서술하면 자동으로 알맞는 방식을 찾아서 처리를 진행한다. 즉 자동으로 먼저 QueryString으로 값을 찾아보고 그것이 없으면 다시 Form으로 찾는다. 만일 QueryString으로 찾아지는 name도 있고 Form으로 찾아지는 name도 있는 경우에는 구체적으로 정확한 입력방식을 지정해주어야 한다.

ServerVariables묶음은 의뢰기가 보낸 모든 HTTP머리부들과 추가항목들을 보관하는것으로서 열람기에 의해 웨브페지요청과 함께 전달되고 요청내용에 대한 추가적인 정보를 포함한다.

ClientCertificate묶음은 의뢰기인증서가 요청되도록 구성된 웨브봉사기에서 인증서를 검사하여 상대방의 신원을 확인하도록 한다.

TotalBytes속성은 의뢰기가 전달한 HTTP요청본문의 전체 바이트수를 보관하는 속성이다.

BinaryRead메소드는 홈의 Post요청에 의해 의뢰기에서 봉사기로 전달된 자료를

얻어내고 이것을 배렬로 저장하는 역할을 하는 메쏘드이다. 이것은 Response.Write로 직접 표시하지 못하고 다만 VBScript의 MidB를 사용하여 출력할수 있으며 실지로는 TotalBytes속성과 함께 쓰인다.

실례:

```
<html>
<body>
<%
intTB = Request.TotalBytes
intInfo = Request.BinaryRead (intTB)
For i = 1 to intTB
Response.Write MidB(intInfo,i,1)
Next
%>
</body>
</html>
```

4.3.2. Response 객체

Response객체는 Request객체에 대응되는 객체로서 봉사기에서 사용자로 어떤 정보를 돌려주는 역할을 하는 객체이다. Response객체는 주로 출력, URL이동(방향바꾸기), 쿠키보내기 등을 담당한다.

Response객체가 가지고있는 기능들은 표 4-5와 같다.

표 4-5. Response의 기능

종 류		설 명
묶음	Cookies	사용자의 열람기에서 보낸 쿠키값 설정
속성	Buffer	페지의 완충기완료 여부
	Expires	열람기에서 고속완충기억기가 없어지기전의 시간
	ContentType	HTTP의 문맥형태(Text/HTML 등)
	ExpiresAbsolute	고속완충된 페지의 끌냄 날자/시간
	Status	봉사기에서 되돌려진 HTTP의 상태값
	isclientconnected	사용자가 봉사기에 련결된 상태의 여부

메쏘드	AddHeader	HTML머리부에 추가,수정
	AppendToLog	봉사기에 Log에 본문을 추가
	Clear	완충된 HTML의 내용삭제
	End	페지처리를 중단하고 현재의 결과를 되돌리게 함
	Flush	완충된 결과를 곧 되돌림
	Write	화면에 변수나 문자렬을 출력
	BinaryWrite	본문을 열람기에 문자설정이 없이 출력
	Redirect	URL의 이동

Response객체의 Buffer속성은 봉사기에서 생성된 자료를 차례차례 즉시 의뢰기에로 넘겨주겠는가 아니면 모든 스크립트가 다 실행된 다음에 넘겨주겠는가 하는 설정을 진행하는 속성이다.

Buffer속성을 True로 설정하면 봉사기는 현재 페지의 모든 ASP스크립트가 처리될 때까지 또는 Flush나 End메쏘드가 호출될 때까지 의뢰기에 출력내용을 보내지 않는다.

반대로 False로 설정하면 봉사기는 의뢰기에 만들어진 순서대로 차례로 보낸다.

Buffer속성은 스크립트의 언어속성을 제외한 그 어떤 스크립트나 문자렬보다 먼저 선언되여야 하며 만일 언어속성의 선언전에 선언된 경우 오유가 발생한다.

Response객체의 Flush, End, Clear메쏘드들은 완충기관련메쏘드들이다.

Buffer속성이 True인 경우에는 완충기에 모든 코드를 저정하게 되며 저장이 다 된 다음에 의뢰기로 자료를 보낸다. 이것은 완충기가 다 찰동안 기다려야 하는 결함이 있다. 이것을 방지하기 위하여 Flush메쏘드를 리용한다.

Flush메쏘드는 완충기에 현재까지 대기상태로 있던 자료를 의뢰기에 즉시 전송하며 스크립트처리를 계속 진행하는 메쏘드이다. 이 메쏘드는 의뢰기에 전송할 자료량이 많은 경우 완충기에 저장하고있는 중간중간에 강제로 현재까지 완충기에 저장된 자료를 의뢰기에 넘겨줄수 있다.

호출방법은 다음과 같으며 전제조건으로는 Buffer속성이 True이여야 한다.

```
<% Response.Flush %>
```

Clear메쏘드는 완충기에 있는 모든 내용을 삭제한다. 그러나 Flush메쏘드가 사용되였다면 마지막으로 호출한 Flush메쏘드에 의해 완충기에 추가된 정보만을 지우고 그렇지 않은 경우 Header를 제외한 모든 페지의 내용을 지운다. 사용방법은 다음과 같은데 이때 Buffer속성은 반드시 True이여야 한다.

```
<% Response.Clear %>
```

End메쏘드는 현재까지 완충기에 저장된 결과를 의뢰기로 전송하며 End메쏘드가 호출된 이후의 모든 스크립트 명령처리를 중단시킨다. 사용방법은 다음과 같은데 이때 Buffer속성은 반드시 True이여야 한다.

<% Response.End %>

쿠키는 의뢰기에 정보를 저장해두었다가 요청에 의해 봉사기로 보내지는 정보이다. 쿠키에 저장된 정보를 다시 의뢰기에 보내기 위해서는 Response객체의 묶음인 쿠키를 사용하여야 하며 저장된 쿠키정보를 읽어오기 위해서는 Request객체의 쿠키를 사용하여야 한다.

실례:

```
<% Response.Cookies("UserID") = "admin" %>
UserID : <% = Request.Cookies("UserID") %>
```

봉사기가 의뢰기에 쿠키를 설정하면 봉사기에 새로운 요청을 할 때마다 쿠키가 자동적으로 봉사기에 전송된다. 이때 봉사기는 의뢰기에 저장되여있는 쿠키에 Expires속성을 사용하여 쿠키의 끝나는 날자를 지정할수 있다. 이것은 봉사기가 의뢰기에 쿠키를 저장시킬 때 설정해주어야 한다. 물론 의뢰기도 봉사기로부터 쿠키를 거절할수 있는 능력이 있다.

쿠키는 자기의 속성들을 지정하지 않은 경우 사용자가 열람기를 끌 때 저장되여있는 정보를 모두 잃어버린다.

형식은 다음과 같다.

```
Response.Cookies(cookie)[(key)|.attribute]=value
```

여기서 cookie는 쿠키의 이름이고 key는 선택적인 매개 변수로서 이것을 지정하면 쿠키의 값은 value로 설정된다. 그리고 attribute는 쿠키자체에 대한 정보를 지정한다. 쿠키에 대한 정보는 다음과 같다.

표 4-6. 쿠키의 정보들

이 름	설 명
Domain	쓰기전용이다. 지정된 쿠키는 이 령역에 대한 응답으로 보내여진다.
Expires	쓰기전용이며 쿠키가 끝나는 날자이다. Session(대화접속)이 완료된 후 의뢰기의 디스크에 쿠키가 저장되게 하려면 이 날자를 반드시 설정해야 한다. 이 속성을 현재 날자이후의 날자로 설정하지 않으면 대화접속이 완료될 때 쿠키도 끝난다.
HasKeys	읽기전용이다. 쿠키가 키를 포함하는가 하는 여부를 지정한다.

경로	쓰기전용이며 지정해주면 쿠키가 지정된 경로에 대한 요청에만 보내진다. 이 속성을 설정하지 않으면 응용프로그람경로가 사용된다.
Secure	쓰기전용이다. 쿠키보안여부를 지정한다.

쿠키는 의뢰기에 파일로 저장되여있기때문에 다른 봉사기에서 접근할수 있다. 이런 문제점을 해결하기 위하여 쿠키에 저장된 정보에 접근할수 있는 봉사기를 제한할수 있다. 또한 정보류출이 크게 문제시되지 않는 정보인 경우 쿠키를 사용하여 봉사기의 부담을 덜어주는것도 좋은 방법이다.

아래에 쿠키를 사용하여 사용자의 방문회수와 방문시간을 저장하였다가 다시 방문하였을 때 쿠키값을 알려주는 실례를 보여주었다.

실례:

```
<html>
<head></head>
<body>
<%
Response.Cookies("UserName") = "최영남"
Response.Cookies("LastVisit") = Now
LastVisit = Request.Cookies("LastVisit")
If Request.Cookies("Visit") = "" Then
   Response.Cookies("Visit") = 0
Else
   Response.Cookies("Visit") = Request.Cookies("Visit") + 1
End If
%>
UserName : <% = Request.Cookies("UserName") %>
<br>
마지막 방문 시간 : <% = LastVist %>
<br>
방문 회수 : <% = Request.Cookies("Visit") %>
</body>
</html>
```

Expires속성은 열람기에 림시 기억된 페지가 끝날 때까지의 기간을 지정하는 역할을 하며 페지가 끝나기전에 사용자가 같은 페지로 돌아오면 기억된 페지가 표시된다.

즉 설정한 시간이 경과되기 전에 사용자가 같은 페지를 방문하면 기억기에 저장된 페지가 나타난다. 만일 시간 값을 《0》으로 주면 이 페지는 기억기에 그 내용이 저장되지 않기때문에 항상 사용자에게 페지의 내용을 새로 보여줄수 있다.

형식:

```
Response.Expires [= number]
```

여기서 number는 페지가 끝나는 시간을 분으로 나타낸 값이다. Asp파일이 Response.Expires를 호출하면 IIS는 봉사기의 시간을 나타내는 HTTP머리부를 작성하며 의뢰기의 체계시간이 봉사기의 체계시간보다 더 빠른 경우 매개 변수를 0으로 설정하면 페지가 즉시 끝나지 않는다.

Response.ExpiresAbsolute속성을 사용하여 페지를 즉시 끝낼수 있다. 그것은 Expires속성에 부의 값을 설정하는것으로 실현할수 있다. 례를 들어 아래와 같이 설정하면

<% Response.Expires = -1 %>

응답이 즉시 완료된다.

한 페지에서 Response.Expires를 여러번 호출하면 봉사기는 가장 짧은 기간을 사용한다.

ExpiresAbsolute속성 역시 Expires속성과 마찬가지로 열람기에 림시 기억된 페지를 끝내는 역할을 한다. 하지만 ExpiresAbsolute속성은 열람기에 림시 기억된 페지가 끝나는 날자 및 시간을 지정할수 있다는 점이 그 차이점이다. 즉 지정한 완료날자 및 시간이 되기전에 사용자가 동일한 페지로 돌아가면 기억된 페지가 나타난다. 그리고 시간을 지정하지 않으면 해당 날자의 지정에 페지가 끝나며 날자를 지정하지 않으면 스크립트가 실행된 날의 설정된 시간에 페지가 끝난다.

형식은 다음과 같다.

```
Response.ExpiresAbsolute [= [date] [time]]
```

여기서 date는 날자를 지정하며 time은 시간을 지정한다. Expires머리부에 보내는 날자값은 RFC-1123날자형식을 따르며 시간은 Expires머리부에 보내기전에 GMT값으로 전환된다.

주의해야 할 점은 이 속성을 페지에 두번이상 지정하면 제일 먼저 지정한 날자나 시간이 사용된다는것이다.

실례:

2007년 5월 31일 오후 1시 30분부터 15초후에 페지가 끝나도록 지정하자.

<% Response.ExpiresAbsolute=#May 31,2007 13:30:15# %>

date값은 유효한 값이여야 하며 반드시 #과 #사이에 값을 넣어야 한다. 그리고 날자의 값을 페지작성한 이전 날자값으로 설정하면 이것은 《Response.Expires = 0》 으로

설정한것과 같다.

ContentType속성은 응답에 대한 HTTP형식을 지정한다. 즉 HTTP머리부의 ContentType머리부를 지정할수 있다.

ContentType를 지정하지 않으면 기정값은 text/html이다.

형식:

```
Response.ContentType [= ContentType ]
```

여기서 ContentType는 문서내용의 형식을 설명하는 문자렬이다. 이 문자렬은 대체로 《type/subtype》형식이다.(여기서 type은 일반적인 범주이고 subtype는 특정한 문서내용형식이다.)

실례:

```
<% Response.ContentType = "application/x-cdf" %>
<% Response.ContentType = "text/HTML" %>
<% Response.ContentType = "image/GIF" %>
<% Response.ContentType = "image/JPEG" %>
<% Response.ContentType = "text/plain" %>
<% Response.ContentType = "image/JPEG" %>
```

IsClientConnected속성은 Response.Write이후에 의뢰기가 봉사기와의 련결을 해제했는가를 나타내는 읽기전용속성이다. 이 속성을 사용하면 의뢰기가 봉사기에서 련결해제한 상태를 더 잘 알수 있다.

형식은 다음과 같다.

```
Response.IsClientConnected ( )
```

의뢰기요청이 이루어진 시간과 봉사기가 응답한 시간사이에 너무 많은 시간이 걸리면 의뢰기가 스크립트를 처리하기 전에 아직 련결되여있는가를 확인하는것이 좋다. 이때 이 속성을 리용한다.

실례:

```
<%
If Not Response.IsClientConnected Then
        Shutdownid = Session.SessionID
        Shutdown(Shutdownid)
End If
%>
```

의뢰기와의 련결을 확인하고 만약 련결이 끊어졌다면 의뢰기에게 할당한 대화접속자원을 해제한다.

Redirect메쏘드는 열람기가 특정 URL로 이동할수 있게 하는 역할을 한다. 즉 사용자가 Redirect로 지정된 웨브페지를 호출했을 때 열람기는 Redirect가 지정한 새로운 웨브페지로 이동한다.

형식은 다음과 같다.

```
Response.Redirect URL
```

여기서 URL은 열람기가 이동할 주소이다.

이 메쏘드는 페지에 의해 설정된 다른 HTTP머리부를 의뢰기에 보내고 이동할 주소를 포함하는 자동응답본문을 작성한다.

실례:

```
<% Response.Redirect "http://www.microsoft.com" %>
```

련결할 페지(Destination Page)가 같은 URL(내 콤퓨터)내에 있다면 다음과 같이 지정할수도 있다.

```
<% Response.Redirect "index.asp" %>
```

주의해야 할것은 <html> 이후이나 Response.Write와 같은 출력을 하는 코드가 Redirect메쏘드전에 오면 ASP2.0에서는 오유가 발생한다는것이다. 하지만 ASP 3.0에서는 아무런 오유없이 Redirect메쏘드가 실행된다. 왜냐하면 ASP 2.0에서는 Buffer의 기본설정이 false로 되여있지만 ASP 3.0에서는 기본설정이 true로 되여있기때문이다. 그러므로 ASP 2.0에서 Redirect메쏘드를 사용하려면 Buffer설정을 true로 지정해주어야 한다.

그리고 Redirect메쏘드는 프레임을 가진 웨브페지에서 그 목적지를 조절할수 없다.

실례:

```
<%
id = Request("id")
pwd = Request("pwd")
If id = "admin" and pwd = "1234" then
    Response.Redirect "../login/login_ok.asp"
else
    Response.Redirect "../login/login_fail.asp"
end if
%>
```

4.3.3. Global.asa 파일

Global.asa파일은 사건스크립트를 지정하고 대화접속령역이나 응용프로그람령역을 가지

는 객체를 선언할수 있는 선택적파일로서 사용자에게 표시되는 응용파일이 아니라 응용프로그람에 의해 대역적으로 사용되는 사건정보와 객체를 저장하는 파일이다. 이 파일의 이름은 반드시 Global.asa로 지정하여 응용프로그람의 뿌리등록부에 저장해야 한다.

매개 응용프로그람들은 Global.asa파일을 하나만 가질수 있다. 즉 반드시 존재해야만 하는 파일은 아니지만 지정한 가상등록부아래에 있는 하위등록부들도 모두 웨브응용프로그람들의 일부이기때문에 이 파일의 영향을 받는다.

Global.asa파일은 Application사건, Session사건, 객체선언, 형식적인 서고선언들에 대한 내용을 포함하고있다.

-Application사건

ASP기반응용프로그람은 뿌리등록부와 하위등록부에 들어있는 파일들로 이루어져있다.

응용프로그람은 사용자가 웨브페지를 해당 응용프로그람에서 처음 열 때 시작되고 봉사기가 완료될 때 끝난다. 응용프로그람에는 Application_OnStart사건과 Application_OnEnd사건이 있다. Global.asa파일에서는 이러한 사건들에 대하여 스크립트를 지정할수 있다. 봉사기는 응용프로그람이 시작되면 Global.asa파일을 검사하여 Application_OnStart사건스크립트를 처리하고 응용프로그람이 끝나면 Application_OnEnd사건스크립트를 처리한다.

우에서 설명한것처럼 응용프로그람(웨브사이트)이 처음 시작될 때 Application_OnStart사건이 발생하고 웨브봉사기가 완료될 때 Application_ OnEnd사건이 발생한다. 그러므로 이때 처리해야 할 사항들을 Global.asa에 넣어주면 된다.

Application_OnStart와 Application_OnEnd사건에 대한 좀 더 구체적인 내용을 아래에서 보기로 하자.

- Application_OnStart

Application_OnStart사건은 첫번째 대화접속이 만들어지기 전에 즉 Session_OnStart사건이 발생하기 전에 발생한다. 이것은 Application과 Server가 제공하는 객체만이 사용할수 있다. Application_OnStart사건스크립트에 있는 Session, Request, 또는 Response객체를 참조하면 오유가 발생한다.

형식은 다음과 같다.

```
<script language=ScriptLanguage runat=Server>
Sub Application_OnStart
...
End Sub
</script>
```

여기서 ScriptLanguage는 사건스크립트를 작성하는데 사용할 스크립트언어를 지정한다. VBScript나 JavaScript와 같은 지원되는 스크립트언어를 지정할수 있다. 두개이상의 사건이 동일한 스크립트언어를 사용하면 하나의 <script>꼬리표로 묶을수 있다.

Application_OnStart에서는 Application.Lock메쏘드와 Application.Unlock메쏘드를 사용할 필요가 없다. 이 메쏘드는 응용프로그람을 시작하는 첫번째 대화접속에서 한번만 호출할수 있다.

실례:

```
Sub Application_OnStart
Application("NumberofVisitors") = 0
End Sub
```

- Application_OnEnd

Application_OnEnd사건은 응용프로그람이 완료될 때 Session_OnEnd사건다음에 발생한다.

Application_OnEnd사건에서 MapPath메쏘드를 호출할수 없다.

형식은 다음과 같다.

```
<SCRIPT LANGUAGE=ScriptLanguage RUNAT=Server>
   Sub Application_OnEnd
   ...
   End Sub
</SCRIPT>
```

여기서 ScriptLanguage는 사건스크립트를 작성하는데 사용할 스크립트언어를 지정한다. VBScript나 JavaScript와 같은 지원되는 스크립트언어를 지정할수 있다.

- Session사건

이전에 대화접속을 가지지 않았던 사용자가 응용프로그람에서 웨브페지를 열면 웨브봉사기가 자동으로 대화접속을 만든다. 봉사기는 대화접속이 시간을 초과했거나 Abandon메쏘드를 호출할 때 대화접속을 끝낸다.

Session에는 두개의 사건, 즉 Session_OnStart사건과 Session_OnEnd사건이 있다. Global.asa파일에 이러한 사건들에 대한 스크립트를 지정할수 있다.

대화접속(Session)이 시작되면 봉사기는 Global.asa파일을 검사하여 Session_OnStart사건스크립트를 처리한다. 이 스크립트는 사용자가 요청한 웨브페지보다 먼저 처리된다. 대화접속(Session)이 끝나면 봉사기는 Session_OnEnd사건스크립

트를 처리한다. 대화접속을 끝내려면 Session객체의 Abandon메쏘드를 호출하거나 Timeout속성을 사용한다.

- Session_OnStart

Session_OnStart사건은 봉사기가 새 대화접속을 만들 때 발생한다. 봉사기는 요청된 페지를 실행하기전에 이 사건을 처리한다.

기본제공객체(Application,ObjectContext,Request,Response, Server, Session)들을 사용할수 있으며 Session_OnStart사건스크립트에서 참조할수 있다.

형식은 다음과 같다.

```
<script language=ScriptLanguage runat=Server>
   Sub Session_OnStart
   ...
   End Sub
</script>
```

여기서 ScriptLanguage는 사건스크립트를 작성하는데 사용할 스크립트언어를 지정한다. VBScript나 Javascript와 같은 지원되는 스크립트언어를 지정할수 있다. Session_OnStart사건이 Redirect메쏘드나 End메쏘드에 대한 호출을 포함하고있으면 Session객체는 지속되여도 봉사기는 Global.asa파일에 들어있는 스크립트와 Session_OnStart사건을 일으킨 파일에 있는 스크립트의 처리를 중지한다.

실례로 Session_OnStart사건에 들어있는 Redirect메쏘드를 호출하여 사용자들이 항상 특정 웨브페지의 대화접속을 시작하도록 할수 있다. 사용자가 응용프로그람을 실행하면 봉사기는 그 사용자용대화접속을 만들고 Session_OnStart사건을 처리한다. 이 사건에 스크립트를 포함시켜 사용자가 열어놓은 페지가 시작페지인가를 확인하고 시작페지가 아니면 Response.Redirect메쏘드를 호출하여 사용자를 시작페지로 보낼수 있다.

Redirect메쏘드를 호출한 다음에 Session_OnStart사건스크립트는 실행되지 않는다. 따라서 사건스크립트에서는 Redirect메쏘드를 마지막에 호출해야 한다.

실례:

```
<script runat=Server language=VBScript>
Sub Session_OnStart
    startPage = "/MyApp/StartHere.asp"
    currentPage = Request.ServerVariables("SCRIPT_NAME")
    If strcomp(currentPage,startPage,1) then
        Response.Redirect(startPage)
    End If
```

```
End Sub
</script>
```

우의 실례는 쿠키를 지원하는 열람기에서만 실행된다. 쿠키를 지원하지 않는 열람기는 SessionID쿠키를 반환하지 않기때문에 봉사기는 사용자가 페지를 요청할 때마다 새 대화접속을 만든다. 따라서 봉사기는 각 요청에 대해 Session_OnStart스크립트를 처리하고 사용자를 시작페지로 보낸다. 거점에서 쿠키를 지원하는 열람기를 필요로 한다는 알림통보문을 시작페지에 표시하는것이 좋다.

실례:

```
<script language=VBScript runat=Server>
  Sub Session_OnStart
     Response.Redirect "http:/server/app/StartHere.asp"
  End Sub
</script>
```

첫번째 실례를 보면 일단 웨브거점의 시작페지(/MyApp/ StartHere.asp)를 지정한 후 대화접속이 련결될 때의 페지(대화접속을 가지지 않았던 사용자가 처음 열려고 시도하는 페지)와 비교한다. 이렇게 시작페지와 사용자가 접근하려는 페지를 문자렬비교하여 일치하지 않으면 원래 지정한 시작페지로 이동시키는 실례이다. StrComp함수는 문자렬의 비교결과를 나타내는 함수이므로 《strcomp(currentPage, startPage,1)》는 currentPage와 StartPage의 문자렬비교를 진행한다.

- Session_OnEnd

Session_OnEnd사건은 대화접속이 취소되거나 시간이 초과될 때 발생한다. 봉사기의 기본객체중에서 Application, Server, Session 객체만 사용할수 있다.

Session_OnEnd스크립트에서 MapPath메쏘드를 호출할수 없다.

형식은 다음과 같다.

```
<script language=ScriptLanguage runat=Server>
   Sub Session_OnEnd
   ...
   End Sub
</script>
```

여기서 ScriptLanguage는 사건스크립트를 작성하는데 사용할 스크립트언어를 지정한다. VBScript나 JavaScript와 같은 지원되는 스크립트언어를 지정할수 있다.

- 객체선언

이 항목을 정확히 리해하기 위해서는 봉사기부분품생성방법에 대해 알고있어야

한다.

일반적으로 ASP에서는 봉사기부분품을 사용하기 위하여 Server객체의 CreateObject메쏘드와 고유의 프로그람식별자(ProgID)를 사용하여 객체의 실체를 생성한다.

실례로 Set objConn=Server.CreateObject("ADODB.Connection")에서처럼 objConn은 객체변수이며 ADODB.Connection이 프로그람식별자이다.

이렇게 생성된 객체변수를 리용하여 그 객체의 속성 및 메쏘드를 사용할수 있다.

실례:

```
objConn.Open
```

메쏘드를 리용하는 방법외에 <object>꼬리표를 사용하여 객체의 실체를 생성할수도 있다.

실례:

```
<object runat="SERVER" scope="PAGE" ID="objConn"
progid="ADODB.Connection">
</object>
```

여기서 scope속성에는 객체의 범위를 지정한다. 우의 실례에서 "PAGE"로 지정되였기때문에 객체를 생성한 ASP페지에만 사용할수 있다. 만약 사용범위를 Session이나 Application까지 넓히려고 한다면 "Session", "Application"선택항목을 사용하면 된다.

보통 "PAGE"선택항목밖에 사용하지 못하지만 <object>꼬리표를 Global.asa안에 넣는다면 "Session", "Application"선택항목을 사용할수 있다. progid대신 객체의 classid를 사용할수도 있지만 잘 쓰이지는 않는다. 확장된 <object>꼬리표를 사용하여 Global.asa파일에 대화접속령역이나 응용프로그람령역을 가지는 객체를 만들수 있다. 이 꼬리표는 독립적이며 모든 <script>꼬리표밖에 존재한다.

Global.asa파일에서 선언된 객체는 봉사기가 해당 객체를 호출하는 스크립트를 처리할 때까지 만들어지지 않는다. 따라서 필요한 객체만 만들기때문에 자원이 절약된다.

형식은 다음과 같다

```
<object runat=Server scope=Scope ID=Identifier {progid="progID"|classid=
"ClassID"}>
...
 </object>
```

여기서 Scope는 객체의 령역을 지정한다. Global.asa파일에서 Scope는 Session이나

Application으로 설정된다. 그리고 Identifier는 객체에 대한 이름을 지정한다. ProgID는 클라스식별자와 련관된 식별자이며 ProgID나 ClassID는 <object>꼬리표안에서 설정한다. ProgID형식은 《[Vendor.]Component[.Version]》이다. ClassID는 COM클라스객체의 고유식별자를 지정한다.

실례:

```
<object runat=Server scope=Session ID=MyConnection
progid="ADODB.Connection">
    REM Object Script
</object>
<object runat=Server scope=Session ID=MyConnection
classid="Clsid:8AD3067A-B3FC-11CF-A560-00A0C9081C21">
    REM Object Script
</object>
```

실례의 첫 부분은 ProgID의 매개 변수를 사용하여 Session객체 MyConnection을 만드는것이고 둘째 부분은 ClassID의 매개 변수를 사용한것이다.

Global.asa파일에서 선언된 객체는 응용프로그람에 있는 모든 스크립트가 사용할수 있다.

실례:

```
---Global.asa---
<object runat=Server scope=Session ID=MyAd
progid="MSWC.AdRotator">
</object>
 ---SOME.ASP---
<%= MyAd.GetAdvertisement("/ads/adrot.txt") %>
```

응용프로그람의 모든 페지에서 MyAd객체를 참조할수 있다. 봉사기부분품의 객체생성여부를 검사하기 위해서는 IsObject()함수(객체가 정상적으로 생성되였다면 True를, 객체생성이 실패했다면 False를 반환)를 리용한다.

즉 우의 실례에 다음과 같은 코드를 추가하면 객체생성여부를 확인할수 있다.

```
<%
If IsObjcet(MyAd) Then
Else
End If
%>
```

- 형식서고 선언

ADO객체에는 속성과 메쏘드에서 사용되는 미리 정의된 많은 상수들이 있다. 이 상수들은 #include문을 리용하여 ASP페지안에 포함시킨다.

실례:

```
< !-- #inlcude File = "adovbs.inc" -->
```

이 방법은 adovbs.inc파일에 정의되여있는 모든 상수를 ASP페지에 포함하기때문에 ASP페지의 용량이 커진다는 결함이 있다. 그래서 더 좋은 방법인 형식서고선언을 리용한다. 형식서고선언은 상수들을 별도의 파일로 포함하지 않고 바로 사용할수 있게 한다.

ASP페지안에 형식서고를 선언하면 포함된 페지내에서는 ADO상수들을 자유롭게 사용할수 있으며 Global.asa파일안에 포함시킨다면 응용프로그람안에 있는 모든 페지에서 상수들을 사용할수 있다.

형식서고를 선언하려면 Global.asa파일에서 <metadata>꼬리표를 사용한다. 실례로 ADO형식서고를 선언하려면 다음 명령문을 사용한다.

```
< !--metadata name="Microsoft ActiveX Data Objects 2.5 Library" type="TypeLib" uuid="{00000205-0000-0010-8000-00AA006D2EA4}"-->
```

또는 형식서고의 UUID(Universal Unique Indentifier)를 사용하는 대신 파일경로를 사용할수도 있다.

```
< !-- metadata type="typelib" file="c:\program files\common files\system\ado\msado15.dll"-->
```

이렇게 ADO형식서고를 선언한 다음에는 Global.asa파일을 포함한 모든 응용프로그람 *.asp파일에서 ADO상수를 사용할수 있다.

다음 실례에서 adOpenKeyset와 adLockOptimistic은 ADO상수이다.

실례:

```
<%
Set rstCustomerList = Server.CreateObject("ADODB.Recordset")
rstCustomerList.ActiveConnection = cnnPubs
rstCustomerList.CursorType = adOpenKeyset
rstCustomerList.LockType = adLockOptimistic
%>
```

#include대신 <metadata>꼬리표를 사용하여 웨브응용프로그람의 성능을 향상시킬수도 있다.

또한 상수를 직접 정의할수도 있다. VBScript에서는 Const문을 사용하고 JavaScript에서는 var문을 사용하여 상수값을 변수에 할당한다. 여러 .asp파일에서 동일한 상수를 사용하려면 상수정의를 별도의 파일에서 진행하고 해당 상수를

사용하려는 .asp파일에 그 파일을 포함하면 된다.

하나의 가상등록부의 뿌리에 존재하는 Glodbal.asa파일과 다른 가상등록부의 뿌리에 존재하는 Global.asa파일사이에는 호상관계가 없다.

4.3.4. Application 객체

Application객체는 웨브봉사기에 접속하는 모든 사용자들이 공유할수 있는 정보(변수, 배렬 혹은 객체)를 제공하는 객체이다. 즉 이 Application객체를 사용하여 해당 ASP기반 응용프로그람을 작성하는 모든 사용자가 정보를 공유하도록 할수 있다. 따라서 Application객체는 모든 사용자들에게 적용되는 간단한 자료를 저장하거나 자료기지 접속과 같이 모든 사용자들에게 사용되는 객체의 참조를 저장하는데 사용된다.

일반적으로 ASP기반 응용프로그람은 가상등록부 및 하위등록부에 있는 모든 파일들이 *.asp로 정의되여있다.

여기서 주의해야 할 점은 Application객체는 여러 사용자가 공유할수 있기때문에 여러 사용자가 한 속성을 동시에 변경할수 없게 해야 한다는것이다.

Application객체가 가지고있는 메쏘드와 사건은 다음과 같다.

표 4-7. Application 객체의 메쏘드와 사건

종 류		설 명
묶 음	Contents	스크립트명령을 통해 Application에 추가된 모든 항목을 포함
	StaticObjects	<object>로 Application에 추가된 모든 객체를 포함
메쏘드	Contents.Remove	Application객체의 묶음에서 선택한 항목을 삭제
	Contents.RemoveAll	Application객체의 묶음에서 항목전체를 삭제
	Lock	다른 사용자가 Application객체의 속성을 수정하지 못하게 함
	UnLock	다른 사용자가 Application객체의 속성을 수정할수 있게 함
사 건	Application_OnStart	Application이 처음 시작했을 때 한번 발생
	Application_OnEnd	Application이 끝날 때 발생

Application객체에 변수 및 객체를 추가하게 되면 모든 사용자들이 이것들을 공유하여 사용할수 있다.

먼저 변수를 Application객체에 추가하는 방법에 대하여 보기로 하자

실례:

```
<%
Application("greeting")= "우리 나라"
Application.Contents("num")=25
%>
```

이렇게 선언된 변수명과 값은 Application객체의 contents묶음에 저장되며 모든 사용자들은 이 변수를 공유할수 있다. Application변수들은 전체 사용자가 공유하는 변수이므로 우의 실례는 어떤 사용자가 실행하더라도 모두 같은 결과를 보여준다.

다음으로 객체를 추가하는 방법을 실례를 들어 보기로 하자.

실례:

```
<%
Set Application("Obj1")=Server.CreateObject("MyComponent")
%>
```

그다음 아래의 스크립트와 같이 웨브페지에서 MyComponent의 메쏘드와 속성을 참조할수 있다.

```
<% Application("Obj1").MyObjMethod %>
```

또는 객체를 국부변수에 할당하여 다음과 같이 리용할수도 있다.

```
<%
Set MyLocalObj1=Application("Obj1")
MyLocalObj1.MyObjMethod
%>
```

여기서 주의해야 할것은 기본제공객체들은 Application객체에 추가할수 없다는것이다.

이렇게 Server.CreateObject메쏘드를 사용하여 변수에 구성요소실체를 할당하면 해당변수는 Contents묶음의 구성원소로 된다. 그러나 <object>로 봉사기부분품의 객체를 생성하고 이것을 Application변수에 할당하면 그 변수는 StaticObject묶음의 구성원소로 된다.

Application.Contents나 Application.StaticObject는 Request객체나 Response객체와 마찬가지로 Application("변수이름")형태로 사용할수 있다. 이렇게 추가한 항목들을 Remove나 RemoveAll를 리용하여 묶음에서 삭제할수 있다.

Application변수는 모든 대화접속에서 공통적으로 사용되는 변수들이다. 따라서

하나의 대화접속에서 Application변수를 변경하면 이 변수를 리용하고있는 다른 대화접속에서 문제가 발생하므로 이런 현상을 없애자면 다른 대화접속에서 이 변수에 대한 변경이 완료될 때까지 사용을 하지 못하도록 하여야 한다. 이 기능을 실현하는것이 바로 Lock메쏘드이다.

반대로 UnLock메쏘드는 Application객체에 저장된 변수를 다른 의뢰기들에서 수정할수 있도록 한다. 이 메쏘드를 호출하지 않으면 *.asp파일이 끝나거나 시간이 초과될 때 웨브봉사기는 Application객체의 잠금을 해제한다.

실례:

```
<%
Application.Lock
Application("NumVisits") = Application("NumVisits") + 1
Application("LastVisited") = Now()
Application.Unlock
%>
```

실례에서 보는바와 같이 Lock메쏘드는 한번에 여러 의뢰기들이 NumVisits변수에 접근하지 못하도록 한다. 만일 Application객체가 잠겨있지 않으면 여러 의뢰기가 동시에 NumVisits변수를 증가시키려고할것이다. 그리고 UnLock메쏘드는 잠긴 객체를 해제하여 그다음 의뢰기가 NumVisits변수를 증가시키도록 한다.

Lock메쏘드를 여러번 호출하는 경우 Application객체의 잠금을 완전히 해제하자면 UnLock를 같은 수만큼 호출하여야 한다. 이렇게 하지 않으면 스크립트가 실행완료될 때까지 Application객체의 잠금이 유지된다.

4.3.5. Session 객체

대화접속(Session)은 사용자와 콤퓨터 또는 두대의 콤퓨터사이의 활성화된 접속을 의미하며 이 대화접속에 관한 모든것을 처리하는것이 바로 Session객체이다.

성공적인 웨브응용프로그람개발에 있어서 어려운 문제들중의 하나가 사용자가 응용프로그람의 여러페지들사이로 이동할 때 접속 또는 대화접속에 대한 사용자정보를 유지하고 관리하는것이다.

HTTP는 상태를 구별하지 않는 통신규약이며 이것은 웨브봉사기가 페지에 대한 매 HTTP요청을 독립적인 요청으로 취급한다는 의미이다. 즉 이전의 요청이 발생한지 불과 몇초 후에 또 새로운 요청이 발생하면 봉사기에는 이전 요청에 대한 정보가 남아있지 않는다.

ASP는 대화접속정보의 관리문제에 대한 방도를 제공한다. 봉사기에 의해 생성된 특정한 사용자식별자(ID)와 ASP Session객체를 사용하여 현재 방문한 매 사용자들을 식별하는 지능적인 응용프로그람을 만들고 정보를 수집하면 응용프로그람은 이러한 정보

를 사용하여 사용자 기본설정과 선택을 추적할수 있다.

ASP는 사용자열람기에 저장된 작은 파일인 HTTP쿠키를 사용하여 사용자식별자를 할당한다. 따라서 쿠키를 지원하지 않는 열람기용응용프로그람을 만들거나 사용자가 쿠키를 거부하도록 열람기를 설정한다면 ASP의 Session관리기능을 사용하지 말아야 한다.

우에서 설명한것처럼 비련결 HTTP통신규약의 ASP기반응용프로그람에서도 Session객체를 사용하면 특정사용자 Session에 필요한 정보를 저장할수 있다.

이렇게 Session객체에 저장된 변수는 사용자가 응용프로그람(웨브거점)에서 페지들사이를 이동할 때 잃어버리지 않는다. 즉 이 Session객체는 사용자가 웨브봉사기와 련결이 지속되는동안 계속 유지된다.

일반적으로 대화접속은 응용프로그람에서 아직 대화접속을 가지고있지 않는 사용자가 웨브페지를 요청하면 웨브봉사기는 자동으로 Session객체를 작성하며 봉사기에서 대화접속이 완료되거나 중단될 때 Session객체를 없애버린다. 여기서 주의할 점은 사용자가 열람기를 닫는다고 해서 대화접속이 끝나는것이 아니라는것이다. 현재의 열람기를 완료하고 다른 열람기를 실행한 경우에만 대화접속이 완료된것으로 본다.

또한 대화접속은 쿠키의 일종이며 ASP에서는 의뢰기열람기에 저장된 작은 쿠키를 사용하여 그 사용자에게 고유한 Session ID를 할당하기때문에 쿠키를 지원하지 않는 열람기인 경우 또는 의뢰기가 쿠키를 거부하도록 열람기를 설정한 경우라면 ASP의 Session기능을 사용할수 없다.

Session값은 봉사기에 저장이 되지만 쿠키는 의뢰기에 저장되는 차이가 있다.

Session 객체가 가지고있는 기능들은 다음과 같다.

표 4-8. Session 객체의 기능

종 류		설 명
속성	CodePage	기호넘기기에 사용되는 코드페지
	LCID	대역적식별자
	SessionID	사용자에 대한 Session ID를 반환
	Timeout	대화접속이 유지되는 시간(분으로 표시)
메쏘드	Abandon	Session객체를 없애고 자원을 해제
	Contents.Remove	Contents에서 항목을 삭제
	Contents.RemoveAll	Contents에서 모든 항목을 삭제
사건	Session_OnStart	새 대화접속을 만들 때 발행
	Session_OnEnd	대화접속이 끝날 때 발생

대화접속의 시작은 Application에 접속하는 순간, 즉 열람기가 웨브봉사기에 처음

페지를 요구했을 때이며 이렇게 대화접속이 시작되는 순간 ASP는 global.asa의 Session_OnStart에 있는 스크립트를 시작한다. 만약 정의된 스크립트가 없다면 아무것도 실행하지 않는다.

대화접속은 다음과 같은 경우에 완료된다.

- 다른 페지에로의 이동이 없이 한 페지에만 머물러서 Session.Timeout이 초과된 경우
- Session.Abandon이 호출된 경우
- 사용자가 열람기를 닫고 다른 열람기를 실행한 경우
- Global.asa파일을 편집한 후 저장했을 경우
- 웨브봉사기가 완료된 경우

사용자들은 대화접속에서 사용해야 할 값들을 대체로 Session변수로 저장해두고 리용한다. 즉 사용자마다의 고유한 값을 Session객체에 저장해둔다. 이렇게 Session객체에 저장된 정보는 Session전체에서 사용할수 있다.

실례:

```
<%
Session("FirstName") = "김"
Session("LastName") = "은주"
%>
```

또한 객체를 Session객체에 저장할수 있다.

객체를 Session객체에 저장하고 기본스크립트언어로 VBScript를 사용하는 경우에는 Set열쇠단어를 사용해야 한다.

실례:

```
<% Set Session("Obj1") = Server.CreateObject("MyComponent.class1")
%>
```

그리고 웨브페지에서 MyComponent.class1에 의해 제공된 메쏘드와 속성을 호출하기 위해 다음 명령을 사용한다.

즉

```
<% Session("Obj1").MyMethod %>
```

또는 다음과 같은 명령을 사용한다.

```
<%
Set MyLocalObj1 = Session("Obj1")
MyLocalObj1.MyObjMethod
%>
```

Session령역을 가진 객체를 만드는 다른 방법은 Global.asa파일에서 <object>끄리

표를 사용하는것이다.
그러나 Session객체에 기본제공객체를 저장할수는 없다. 실례를 들어 아래와 같이 프로그람을 작성하면 오유를 발생한다.

실례:

```
<%
Set Session("var1") = Session
Set Session("var2") = Request
Set Session("var3") = Response
Set Session("var4") = Server
Set Session("var5") = Application
%>
```

Session객체에서 사용자의 기본설정을 저장한 다음 이 기본설정에 의해 사용자에게 반환할 페지를 결정할수 있다.

실례:

```
<% If Session("Page") = "Low" Then %>
이것은 책상입니다.
<% Else %>
이것은 의자입니다.
<% End If %>
```

Session객체의 Contents에는 Session에 대해서 저장(<object>끄리표를 사용하지 않고 저장)된 모든 변수가 포함되여있다.

Contents의 Remove메쏘드를 사용하면 대화접속에 추가된 변수중 임의의것을 선택하여 제거할수 있으며 RemoveAll메쏘드를 사용하면 대화접속에 저장된 모든 변수를 완전히 제거할수도 있다.

실례:

```
<%
If Session.Contents("Purchamnt") <= 75 then
Session.Contents.Remove("Discount")
End If
%>
<% Session.Content.RemoveAll() %>
```

- Timeout속성

Timeout속성은 Session객체에 지정된 제한시간을 분단위로 지정한다.

사용자가 이렇게 지정한 제한시간내에 페지를 새로 고치거나 요청하지 않으면

대화접속은 끝난다.
형식은 다음과 같다.

```
<% Session.Timeout = 20 %>
```

- Abandon메쏘드

Abandon메쏘드는 Session객체에 저장된 모든 객체를 삭제하고 자원을 해제한다. 이 Abandon메쏘드를 호출하지 않으면 대화접속이 시간초과될 때 봉사기가 이 객체를 삭제한다.

형식은 다음과 같다.

```
<% Session.Abandon %>
```

Abandon메쏘드를 호출하면 현재 페지의 모든 스크립트명령이 처리될 때까지는 실제로 삭제되지 않는다. 즉 Abandon메쏘드를 호출한 페지에서는 Session객체에 저장된 변수에 접근할수는 있지만 다른 웨브페지에 있는 이 변수에는 접근할수 없다는 의미이다.

실례를 들면 다음 스크립트에서 세번째 행의 실행결과는 Mary가 인쇄된다. 그 리유는 봉사기가 스크립트처리를 끝낼 때까지는 Session객체가 없어지지 않기때문이다.

실례:

```
<%
Session.Abandon
Session("MyName") = "Mary"
Reponse.Write(Session("MyName"))
%>
```

만일 다른 웨브페지에서 MyName변수에 접근하면 이 변수는 비여있다. 그 리유는 우의 실례가 포함된 페지가 다 처리되였을 때 MyName이 이전 Session객체와 함께 없어졌기때문이다.

그리고 어떤 대화접속을 취소한 후에 다음 웨브페지를 열면 봉사기는 Session객체를 새로 작성하여 새로 작성된 Session객체에 변수와 객체를 저장한다.

4.3.6. Server 객체

ASP가 웨브응용프로그람을 제작하는데 필요한 모든 기능을 완벽하게 제공하는것은 아니다. 그러나 Server객체를 리용하면 ASP에서 지원하지 않는 기능을 확장시킬수 있다.

Server객체는 ScriptTimeout속성과 CreateObject, Execute, GetLastError, HTMLEncode, URLEncode, Transfer, MapPath의 메쏘드들을 가지고있다. 이 메쏘드들중에서 제일 많이 사용하는 메쏘드는 CreateObject이다.

- ScriptTimeout속성

이 속성은 스크립트가 실행될수 있는 최대시간을 지정하는 속성이다. 즉 스크립트가 ScriptTimeout속성에 정해놓은 시간을 초과하면 자동적으로 실행이 중단되게 된다.

ASP응용프로그람에서 실행도중 무한순환에 빠지게 되면 봉사기성능이 떨어지며 최악의 경우에는 봉사기가 실행되지 않는 경우가 있는데 이런 경우 ScriptTimeout속성을 리용하여 웨브응용프로그람을 강제로 중단시킬수 있다.

형식은 다음과 같다.

```
Server. ScriptTimeout=NumSeconds
```

여기서 NumSeconds는 봉사기에서 스크립트가 실행될수 있는 최대시간을 초로 지정하는 값이다. 기정값은 90초이다.

- CreateObject메쏘드

CreateObject메쏘드는 봉사기구성요소의 실체를 생성하는데 리용한다.

ASP에서는 유용하게 사용될수 있는것들을 부분품으로 제공하고있는데 이러한 부분품들을 사용하는데 쓰이는것이 바로 이 메쏘드이다.

형식은 다음과 같다.

```
Server.CreateObject(progID)
```

여기서 progID는 작성할 실체의 형식을 지정한다. progID의 형식은 [Vender.]Component[.Version]이다.

Server.CreateObject메쏘드에 의해 작성된 실체는 페지령역을 가진다. 즉 현재 ASP페지의 처리가 끝나면 이 실체는 자동적으로 없어진다. 그러므로 대화접속령역이나 응용프로그람령역을 가지는 실체를 작성하려면 Global.asa파일에 <object>를 사용하여 Scope속성을 Session이나 Application으로 설정하거나 또는 Session변수나 응용프로그람변수에 실체를 저장하여야 한다.

아래의 실례와 같이 Session실체가 없어질 때 Session변수에 저장된 실체도 없어지므로 대화접속이 제한시간이 되면 Abandon메쏘드가 호출된다.

실례:

```
<% Set Session("ad") = Server.CreateObject("MSWC.AdRotator")%>
```

또한 아래의 실례에서처럼 변수를 Nothing으로 설정하거나 변수를 새 값으로 설정하여 실체를 없앨수도 있다. 실례의 첫행은 ad실체를 적제하는것이고 둘째행은 ad를 문자렬로 바꾼다.

실례:

```
<% Session("ad") = Nothing %>
<% Session("ad") = "some other value" %>
```

ASP에서는 자료기지에 대한 접근을 제공하는 ADO부분품이 있다. 이 부분품을

사용하기 위하여 우선 객체를 선언하여야 한다. 그리고 이렇게 생성된 객체의 메쏘드나 속성을 리용하여 자료기지와 련동한다.

실례:

```
<% Set Rs= Server.CreateObject("ADODB.Recordset") %>
```

여기서 Set는 생성된 객체를 객체변수에 지정해주는 열쇠단어이다.

- Execute메쏘드

Execute메쏘드는 ASP3.0에 새롭게 추가된 메쏘드로서 *.asp파일을 호출하여 마치 이 파일을 호출한 ASP스크립트의 일부인것처럼 처리하도록 하는 메쏘드이다. 즉 다른 asp파일을 현재의 페지로 불러들여 실행한 후 다음 스크립트를 처리한다.

이 메쏘드는 복잡한 응용프로그람을 작성할 때 개별모듈로 분할하여 처리할 때 리용한다. 필요할 때 호출할수 있는 ASP파일의 서고를 작성해야 한다.

형식:

```
Server.Execute(Path)
```

여기서 Path는 실행할 *.asp파일의 위치를 지정하는 문자렬이다. 절대경로나 상대경로중 하나가 될수 있으며 이 매개 변수에 절대경로를 지정하면 같은 응용프로그람내에 있는 .asp파일에 대해서도 절대경로를 지정하여야 한다.

실례:

```
<html>
<body>
 <%
 Response.Write "ASP2페지 호출하기 전 <br>"
 Server.Execute "./asp2.asp"
 Response.Write "<br> ASP2페지 호출한 후"
 %>
 </body>
 </html>
```

실례:

```
"./asp2.asp"
<html>
<body>
<% Response.Write "ASP2페지이다." %>
</body>
</html>
```

결과는 다음과 같다.

ASP2페지 호출하기 전
ASP2 페지이다.
ASP2 페지 호출한 후

실레에서 보다싶이 Execute메쏘드가 호출된 시점이후의 스크립트는 계속 실행된다.

- Transfer메쏘드

이 메쏘드는 현재까지의 스크립트실행을 끝내고 지정된 파일을 불러들여 실행하는 메쏘드이다. 이 메쏘드가 호출되면 그후의 모든 스크립트처리를 중단하게 된다. Transfer를 호출하면 모든 기본제공객체의 상태정보가 전송된다. 즉 대화접속이나 현재 응용프로그람이 실행되고있는 동안 호출한 페지에서 값이 할당된 변수나 객체에 모두 유지된다.

형식:

```
Server. Transfer(Path)
```

여기서 path는 전송할 파일의 위치이다.

실례:

```
<html>
<body>
<%
Response.Write Session.SessionID
Response.Write "<br>"
Response.Write "같은 Session ID가 전송되겠는가?<br>"
Server.Transfer "./asp2.asp"
Response.Write "이 부분은 출력이 되겠는가?"
%>
```

실례:

```
"./asp2.asp"
<html>
<body>
<% Response.Write Session.SessionID %>
</body>
</html>
```

결과는 다음과 같다.

123456789
같은 Session ID가 전송되겠는가?

123456789

우의 결과를 보다실이 Execute와는 달리 Transfer가 호출된 시점 이후의 스크립트처리가 중단되는것을 확인할수 있다.

- MapPath메쏘드

MapPath는 지정된 상대경로 또는 가상경로를 봉사기상에서 해당하는 실지의 경로로 변환하는 메쏘드이다. MapPath는 변환하는 경로가 유효한지 또는 봉사기에 존재하는가를 확인하지 않는다.

형식은 다음과 같다.

```
Server.MapPath(Path)
```

여기서 Path는 실지의 경로로 변환할 상대경로 또는 가상경로를 지정한다. Path가 사선(/)이나 역사선(\)으로 시작되면 MapPath메쏘드는 Path가 모두 가상경로인것으로 보고 경로를 변환한다. Path가 사선으로 시작하지 않으면 MapPath메쏘드는 처리되고있는 .asp파일의 Path를 상대경로로 보고 경로를 변환한다.

실례:

현재 파일의 절대경로: Server.MapPath(".")

```
<%= Server.MapPath(".")%><br>
```

현재 파일의 상위 등록부의 절대경로: Server.MapPath("..")

```
<%= Server.MapPath("..")%><br>
```

현재 파일의 홈등록부의 절대경로: Server.MapPath("\")

```
<%= Server.MapPath("\")%><br>
```

가상등록부 test의 절대경로: Server.MapPath("\test")

```
<%= Server.MapPath("\test")%><br>
```

- HTMLEncode메쏘드

이 메쏘드는 HTML원천을 열람기에서 그대로 볼수 있게 변환해주는 메쏘드이다.

HTML은 시작꼬리표 《<》와 끝꼬리표 《>》로 구분하는데 HTML꼬리표를 열람기에서 그대로 보려고 하면 《<》기호는 《<》로, 《>》기호는 《>》로 변환해 주면 된다.

형식은 다음과 같다.

```
Server.HTMLEncode( string )
```

여기서 string은 부호화할 문자렬을 지정한다.

- URLEncode메쏘드

URLEncode는 문자렬로 된 정보를 입력받아 URL로 부호화된 형태로 변환하는 메쏘드이다.

형식은 다음과 같다.

```
Server.URLEncode( string )
```

여기서 string은 부호화할 문자렬을 지정한다.

제4절. ASP의 응용

ASP의 응용실례로 현재까지 웨브거점에 대한 사용자들의 방문정형을 보여주는 계수기프로그람을 주었다.

계수기를 실현하기 전에 먼저 환경을 설정하여야 한다.체계는 Window 2000 Server로 하고 Visual InterDev편집기를 설치하여야 한다.

실현할 계수기는 거점을 방문한 현재까지의 총 인원수와 오늘 방문한 수, 그리고 현재 방문중인 인원수를 출력하는 프로그람이다.

먼저 Notepad를 열어서 첫번째 행에 0, 두번째 행에 0, 세번째 행에는 적당한 날자를 입력한 후 《Inet/wwwroot/Project1》에 《MyCounter.txt》로 보관한다. 이 파일은 웨브거점을 방문한 총 방문자수와 오늘 방문한 수를 저장하기 위한 text파일이다.

봉사기가 동작중일 때에는 총방문자수가 Application변수로 저장된다. 봉사기가 동작하지 않을 때에는 Application변수는 없어진다. 그러므로 이러한 경우를 예견하여 봉사기의 동작과는 무관계한 파일로서 총 방문자수를 저장하기 위한 text파일을 만든다

Global.asa파일은 다음과 같다.

```
<Script Language="VBScript" RunAt="Server">
  Sub Application_OnStart
    CFile = Server.MapPath(".")
    CFile = CFile & "\MyCounter.txt"
    Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
    Set objOTF = objFSO.OpenTextFile(CFile, 1, false)
   Application("Total_Counter") = objOTF.ReadLine
    Application("Today_Counter") = objOTF.ReadLine
    Application("Today") = objOTF.ReadLine
    Application("Now_Counter") = 0
    Application("CFile") = CFile
  End Sub
Sub Session_OnStart
    Application.Lock
    Application("Total_Counter") = Application("Total_Counter") + 1
    Application("Now_Counter") = Application("Now_Counter") + 1
```

```
    if Application("Today") = Date() then
        Application("Today_Counter") = Application("Today_Counter") + 1
        Application("Today") = Date()
    else
        Application("Today_Counter") = 1
        Application("Today") = Date()
    end if
    Application.UnLock
     If (Application("Total_Counter") MOD 5) = 0 Then
        Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
        Set objCTF = objFSO.CreateTextFile(Application("CFile"), true)
        Application.Lock
        objCTF.WriteLine(Application("Total_Counter"))
        objCTF.WriteLine(Application("Today_Counter"))
        objCTF.WriteLine(Application("Today"))
         Application.UnLock
      End If
End Sub
Sub Session_OnEnd
      Application.Lock
      Application("Now_Counter") = Application("Now_Counter") - 1
      Application.UnLock
End Sub
Sub Application_OnEnd
      Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
       Set objCTF = objFSO.CreateTextFile(Application("CounterFile"), true)
     Application.Lock
       objCTF.WriteLine(Application("Total_Counter"))
       objCTF.WriteLine(Application("Today_Counter"))
       objCTF.WriteLine(Application("Today"))
     Application.UnLock
End Sub
</Script>
```

우의 실레에서 보여주는바와 같이 이 파일을 만드는데는 먼저 Server객체를

리용하여 MyCounter.txt파일의 가상등록부의 경로를 물리적으로 알아내고 이것을 CFile변수에 할당한다. 여기서 《.》은 현재의 물리적인 등록부를 나타낸다.

다음 FileSystemObject의 객체를 objFSO이름으로 생성하고 OpenTextFile을 리용하여 objOTF라는 이름으로 TextStream객체를 생성한다. 그리하여 CFile 즉 MyCounter.txt파일을 읽기속성으로 연다. 만일 해당 파일이 없으면 오유가 발생한다. 이때 TextStream객체는 지정된 파일을 열고 읽고 쓰며 덧붙이기도 할수 있는 속성과 메쏘드를 제공하는 객체이다.

이것을 리용하여 MyCounter.txt파일에 있는 총 방문자수와 오늘 방문자수 그리고 현재날자를 읽어서 Application변수(Total_Counter, Today_Counter,Today)에 할당한다. 그리고 현재 방문자수는 Now_Counter변수로 선언하여 처음값을 0으로 주며 MyCounter.txt파일의 절대경로를 Application변수인 CFile에 할당한다.

다음으로 새로운 방문자가 들어오면 대화접속상태를 차단(Lock)하여 다른 사용자들이 사용하지 못하게 한 다음 전체 방문자수와 현재 방문자수를 하나 증가시키고 날자가 오늘 날자와 같으면 오늘 방문자수에 1을 더하고 그렇지 않을 경우에는 오늘 방문자수에 1을 할당하여 오늘 방문자수를 구한다.

그 다음 차단을 해제시킨다. 여기서 차단을 사용하는 리유는 변수에 해당 응용프로그람의 모든 사용자들이 접근할수 있으므로 동시에 발생할수 있는 다른 사용자의 접근을 막기 위해서이다.

다음 구문은 총 방문자수를 5으로 나누었을 때 나머지가 0이면 Create TextFile메쏘드로서 TextFile을 덧쓰기속성으로 연 다음 차단상태로 만들고 WriteLine메쏘드로 매개 Application변수들을 덧쓰기한다. 그 리유는 봉사기가 동작중일 때는 모든 변수들이 Application변수로 저장되지만 그렇지 않을 때는 이 변수들이 다 없어지기때문에 이런 경우를 예견하여 봉사기의 동작과는 무관계한 파일로 총 방문자수를 저장하기 위해서이다.

다음 Session_OnEnd부분은 한 사용자의 대화접속이 끝나면 현재 방문자수중에서 하나를 더는 부분이다. 그리고 Application_OnEnd부분은 응용프로그람이 완료될 때 처리하여야 할 내용을 서술한것이다. 즉 응용프로그람이 완료되면 현재까지의 총 방문자수, 오늘 방문자수, 현재 날자를 저장해야 하므로 우의 Session_OnStart부분에서 설명했던 방식으로 MyCounter.txt파일에 저장한다.

이렇게 되면 계수기에서 보여주려는 내용을 모두 서술한것으로 된다.

다음으로 MyCounter.asp파일을 다음과 같이 만들고 열람기에서 확인해본다.

```
<%@ Language=VBScript %>
<html>
<head>
```

```
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</head>
<body>
<%
CFile = Server.MapPath(".")
CFile = CFile & "\MyCounter.txt"
Response.Write CFile
%>
<br>
총 방문자 수 : <%=Application("Total_Counter")%>
<br>
오늘 방문자 수 : <%=Application("Today_Counter")%>
<br>
현재 방문자 수 : <%=Application("Now_Counter")%>
</body>
</html>
```

제 5 장. JSP 언어

제1절. JSP언어의 기초

5.1.1. JSP 와 CGI, Servlet

Java Server Pages(JSP)는 동적인 웨브페지를 쉽게 만들수 있는 방법을 제공하며 JAVA를 그대로 사용할수 있기때문에 더욱 효률적이다. 또한 스크립트방식으로 프로그람을 작성할수 있는 봉사기측스크립트언어로서 편리하고 확장성이 매우 크다.

일반적으로 HTML문서는 HTTP통신규약을 리용하여 단순하게 보여주는 정적문서로서 의뢰기에서 봉사기에 문서를 요구하면 봉사기는 이미 작성되여있는 문서를 그대로 봉사한다. 하지만 동적인 페지를 요구하게 되면서부터 의뢰기와 웨브봉사기, CGI응용프로그람을 결합시켜 동적인 웨브페지를 만들게 되였다. 이 CGI를 받아들인 결과 동적웨브페지에서는 자료기지와 관련되는 많은 봉사와 함께 실시간적으로 제기되는 각종봉사를 진행할수 있게 되였다.

CGI(Common Gateway Interface)는 웨브봉사기와 열람기, 응용프로그람들사이의 일반적인 대면부를 말한다.

일반적으로 봉사기측에서 처리하여야 하는 자료기지작업이나 계산작업을 주로 수행하는 이 CGI응용프로그람은 대면부를 더욱 발전시켜 사용자에게 편리한 형태로 전환되였는데 그 대표적인 례가 바로 JSP와 ASP, PHP이다.

JSP는 종래의 단순한 HTML문서만을 봉사하던 웨브봉사기의 기능을 보다 발전시켜 웨브기반의 프로그람을 처리할수 있도록 만든것이다. 그리고 Servlet를 기반으로 하고있으며 Servlet의 프로그람적인 요소를 발전시켜 사용자가 보다 쉽게 다룰수 있도록 만든 스크립트형식의 프로그람이다.

JSP는 내부적으로 Servlet를 사용하고있으며 스크립트페지는 결국 Servlet로 콤파일되여 봉사요청에 응답한다.

Servlet는 문자그대로 봉사기에서 실행되는 프로그람이다.

봉사기에서는 프로그람을 처리하고 그 결과를 의뢰기에로 전송하는 방식을 사용한다. Servlet가 프로그람준위에서 봉사를 처리한다면 JSP는 스크립트준위에서 봉사를 처리한다고 볼수 있다.

일반적인 CGI응용프로그람과 JSP의 가장 큰 차이점은 프로세스의 생성인가 아니면 스레드의 생성인가하는 문제에서의 차이이다.

CGI응용프로그람은 사용자요청(Request)이 있을 때에 프로세스를 생성한다. 그리고 작업을 다 한 다음에는 프로세스를 끝내게 된다. 그러나 JSP는 사용자요청

(Request)이 있을 때 필요한 클라스를 동적으로 만들고 그 다음 객체를 생성한다. 가상기계가 클라스의 단위로 이루어지기때문에 모든 작업은 클라스형태로 이루어지게 된다. 작업을 처리하는것을 JSP용기라고 부르며 작업을 끝마친 후에도 이 JSP용기에 적재되여있게 된다. 즉 객체의 기억은 공유되며 스레드를 생성하여 작업을 처리하게 된다.

첫 적재시간은 오히려 CGI보다도 시간이 더 걸릴지 모르지만 한번 적재된 다음부터는 빠르게 처리하고 프로쎄스의 관리보다는 스레드의 관리가 더 쉽다는것이 JSP의 우점이다.

또한 CGI에서는 프로쎄스를 생성하고 작업을 마친 뒤 끝나지만 JSP에서는 다시 사용할수 있다는것이 가장 큰 차이점이다.

5.1.2. JSP의 특징

첫째로, 봉사기환경에서 사용하는 스크립트방식의 언어이다.

둘째로, JSP는 Java를 기반으로 하고있기때문에 Java의 우점을 전부 활용할수 있다.

JSP는 사용하는 언어가 Java이다. Java언어는 그 어떤 가동환경이나 어떤 웨브봉사기에서도 사용할수 있다.

셋째로, 의뢰기의 요청이 있을 때 JSP는 하나의 기억을 공유하면서 _jspService메쏘드만을 호출한다.

JSP는 단일스레드로 의뢰기의 요청에 봉사한다. 요청이 있을 때마다 프로쎄스를 생성하는 종래의 CGI응용프로그람과는 달리 JSP에서는 하나의 기억을 공유하면서 봉사되므로 봉사기측의 부하를 덜어준다.

넷째로, JSP는 보여주는 부분과 작업부분을 분리할수 있다. JSP내부에는 보여주는 코드만 작성하고 직접 작업하는 부분은 JavaBean으로 구성하여 분리할수 있다. 이것은 서로 영향을 주지 않고 수정할수 있는 우점을 가지고있다. 또한 Java의 우점인 재사용성을 높일수 있다.

이렇듯 많은 우점들이 있지만 Java자체의 실행에서 성능이 낮으며 JSP를 처음 적재할 때에는 속도가 느리다는 부족점 즉 JSP파일을 Servlet로 변환하고 콤파일하는 과정에 의하여 발생하는 작업부하 등이 문제점으로 되지만 하드웨어성능의 향상과 대용량화에 의하여 이러한 부족점은 줄어들고있다.

5.1.3. JSP의 구조

- JSP의 해설문

이것은 JSP를 실행하는 JSP엔진에 알려주는 설명내용으로서 프로그람실행에는 아무런 영향을 주지 않는 요소이다. 이 해설문은 사용자가 열람기를 리용하여 코드를 볼 때 나타나지 않는다.

형식은 다음과 같다.

```
<%----------해설문------------%>
```

- JSP의 선언문

선언문은 JSP페지에서 사용할 변수나 메쏘드를 선언하는데 사용한다. JSP에서 사용하는 선언방법에는 두가지가 있다.

첫번째 방법은 선언꼬리표를 리용하여 변수나 메쏘드를 선언하는것이고 두번째 방법은 JSP스크립트에서 선언하여 사용하는 방법이다.

JSP에서는 선언꼬리표를 리용하여 같은 자료형의 여러 변수들과 서로 다른 자료형의 변수를 여러개 동시에 선언할수 있다.

형식은 다음과 같다.

```
<%!-----선언할 내용---------%>
```

실례:

```
<%! Int i=0;%>
<%! Int a=0,b,c;%>
<%! Int j=0; string str;%>
```

JSP에서는 변수를 사용하기전에 반드시 변수를 선언해주어야 한다. 선언꼬리표내에서 변수선언의 수는 제한이 없으며 서로 다른 자료형의 변수는 반두점(;)으로 구분하여 선언할수 있다.

선언된 변수는 해당 페지에서만 유효하다. 만일 <%@ include%>꼬리표나 <jsp:include>꼬리표를 사용하여 페지에 포함한 경우 변수는 그 포함된 문서에도 영향을 미친다. 그러나 포함된 문서가 동적으로 생성된 페지라면 변수는 그 문서에 영향을 주지 못한다.

- JSP의 출력문

출력문은 JSP에서 동적인 웨브문서의 생성을 위해 사용한다.

형식은 다음과 같다.

```
<%= 표현식%>
```

출력문은 표현식의 자료형에 관계없이 표현식의 결과를 문자렬로 변환한다.

출력문에서 표현식의 결과를 문자렬로 변환하려면 그 표현식의 결과값은 반드시 문자렬로 변환될수 있는 자료형이여야 한다.

여기서는 반두점(;)을 사용할수 없다.

실례:

<%= "내나라 제일로 좋아" %>

- 스크립트레트(Scriptlet)

JSP에서 코드는 반드시 <% 와 %>안에 서술하여야 한다. 그리고 한문장이 끝나면 그 끝을 반두점으로 알려주어야 한다. 이 반두점을 리용하면 한 행에 두개의 문장을 쓸수도 있다.

JSP스크립트는 보통 Java에서 사용하는 문법과 같다. JSP에서는 일반문서나 HTML문서, 또는 JSP문장요소 등을 직접 사용할수 있다.

5.1.4. JSP 의 동작과정

먼저 사용자의 요청이 있어야 한다. 이 요청에는 JSP페지의 실행만을 요구하는 경우와 JSP페지에 사용자가 요구하는 질문을 보내고 실행을 요구하는 경우가 있다.

사용자는 JSP페지에 요청할 때 JSP가 처리할수 있는 값을 JSP페지에 보낸다. 그 질문을 받은 JSP가 처리할 내용은 간단한 스크립트만을 가지고 실행할수 있는 경우와 JavaBean을 리용하여 실행할수 있는 경우 그리고 자료기지를 같이 동작시켜 그 자료의 처리를 진행하여 결과를 얻어오는 경우가 있다. 여기서 어떤 방법을 쓰는가는 사용자의 설계에 따라서 결정된다.

그 다음 JSP페지가 실행되면 그 실행결과가 얻어지는데 그것을 HTML문서로 변형시켜서 열람기로 보낸다. 그러면 열람기는 그 문서를 사용자가 볼수 있도록 출력한다.

이것이 JSP의 기본동작과정이다.

봉사기측에서 JSP페지를 실행시키고 그 결과를 HTML문서로 만들어서 열람기에게 보내는 일을 하는것이 바로 JSP엔진이다.

그림 5-1에 그 과정을 보여준다.
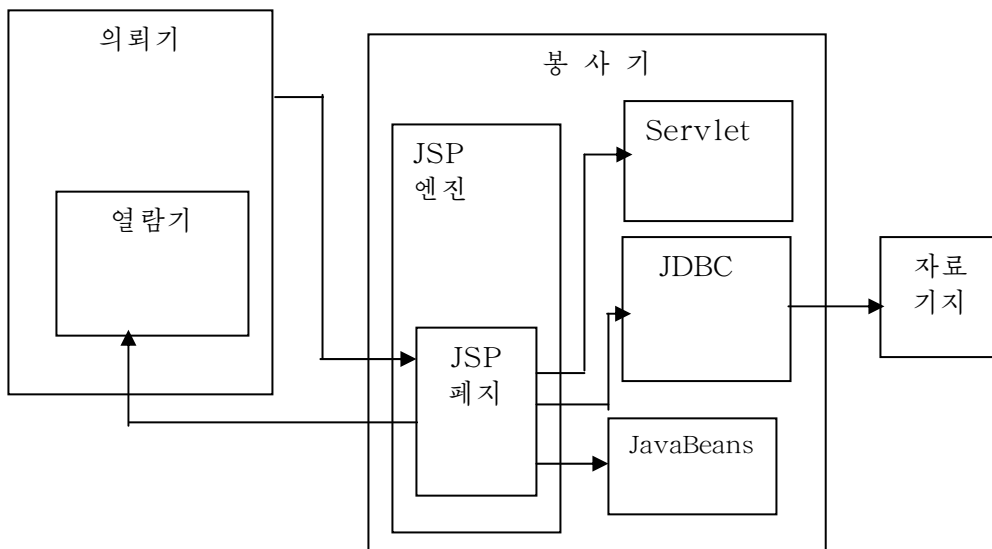
그림 5-1. JSP의 동작과정

제2절. JSP의 문법

5.2.1. 자료형

- 식별자(Identifier)

식별자는 JSP에서 유일하게 식별할수 있는 변수, 상수, 배렬, 클라스, 메쏘드 등의 이름을 말한다. 식별자는 반드시 문자로 시작하여 문자나 수자로 끝나야 한다. 글자수는 한 글자이상이여야 하며 글자수는 제한이 없다. JSP는 식별자를 구분할 때 대소문자를 구분한다.

- 예약어

예약어란 JSP에서 미리 의미를 정하여 사용하는 단어를 말한다. 이 예약어는 JSP에서 미리 정한 단어이므로 식별자로 사용하여서는 안된다.

JSP의 예약어는 다음과 같다.

Abstract	double	int
boolean	else	interface
break	extends	long
byte	final	native
case	finally	new
catch	float	package
char	for	private
class	goto	protected
const	if	public
continue	implement	return
default	import	short
do	instanceof	static

- 자료형

자료형은 JSP에서 사용하는 자료의 류형으로서 자료가 가질수 있는 값, 값범위, 연산규칙 등을 결정하는 역할을 한다.

자료형은 크게 두가지 즉 기본형과 참조형으로 나눈다.

기본형은 모든 언어들이 기본적으로 제공하는 자료형으로서 정수형(int, short, byte, long), 실수형(float, double), 문자형(char), 론리형(boolean) 등이 있다. JSP에서는 가동기반에 상관없이 자료형의 크기가 일정하다.

참조형은 기본형과는 달리 직접적인 값을 저장하고있는 자료형이 아니라 어떤 배렬이나 클라스와 같은 자료의 모임에 대한 참조만을 가지는 자료형을 말한다. 여기에는 배렬형, 클라스형, 대면부형 등이 있다.

- 상수형

상수는 변하지 않고 늘 일정한 수를 의미한다. 상수에는 기본형과 객체, 문자렬이 있는데 기본형의 상수는 정수, 실수, 8진수, 16진수, 문자, true, false 등이 있다.

객체에 대한 유일한 상수는 null이다. 이것은 참조형에서만 사용할수 있다.

문자렬은 괄호안에 임의의 문자를 쓸수 있다.

- 변수

변수란 그 어떤 값을 보관할수 있는 공간이다. 변수를 리용할 때에는 반드시 변수의 선언을 진행해야 한다.

변수선언방법에는 선언꼬리표를 리용하는 방법과 스크립트내에서 서술하는 방법이 있다.

스크립트내에서 선언하는 방법은 다음과 같다.

```
<%
int a;
int b=0;
string str= “내나라 ”
%>
```

JSP프로그람내에서 사용하는 변수는 해당 JSP페지내에서만 유효하다.

즉 JSP문서에서 include를 리용하여 다른 문서를 포함하는 경우라도 그 포함되는 문서가 동적인 문서 즉 또 다른 JSP문서라면 같은 페지라 할지라도 포함된 문서에 영향을 주지 않는다.

- 배렬형

배렬은 같은 자료형의 자료를 여러개 가질수 있는 자료형으로서 각각의 요소들에 그 배렬자료의 첨수를 리용하여 접근할수 있다. 다른 변수들과 마찬가지로 배렬을 사용하기전에 반드시 선언을 하여야 한다. 배렬의 선언식은 자료형이름과 배렬기호, 그리고 배렬이름으로 구성된다.

실례:

```
int[] a;
```

배렬을 선언한 다음에는 배렬을 생성하여야 한다.

실례:

```
a=int[5];
a={1,2,3,4,5}
```

JSP에서는 배렬에 값을 대입할 때 배렬의 첨수를 사용한다. 배렬의 첨수는 0부터

시작한다.

실례:

```
a[1]=5;
int b=a[3];
```

5.2.2. 연산자

- 산수연산자

산수연산자는 수값계산을 위한 연산자로서 이항연산자와 단항연산자로 구분한다.

표 5-1. 산수연산자

기호	역 할
+	더하기
-	덜기 혹은 부의 값을 표시
*	곱하기
/	나누기
%	나머지연산
++	값을 하나 증가
--	값을 하나 감소

여기서 《+》연산자는 문자렬을 련결시키는 기능도 수행한다.

- 관계연산자

두개의 값을 비교하는 연산자로서 연산결과는 참 또는 거짓을 가진다.

표 5-2. 관계연산자

문법	true의 값이 주어진 경우의 의미
x==y	x는 y와 같다.
x!=y	x는 y와 같지 않다.
x<y	x는 y보다 작다.
x<=y	x는 y보다 작거나 같다.
x>y	x는 y보다 크다.
x>=y	x는 y보다 크거나 같다.

- 론리연산자

량변의 값들을 론리적으로 비교하는 연산자이다.

표 5-3. 론리연산자

연산자	의 미
a&&b	a,b가 모두 참이면 참 (a가 거짓이면 b를 계산안함)
a||b	a 또는 b가 참이면 참 (a가 참이면 b를 계산안함)
a&b	a,b가 모두 참이면 참 (a와 b를 모두 계산)
a|b	a 또는 b가 참이면 참 (a와 b를 모두 계산)
!a	a가 참이면 거짓, 거짓이면 참

- 조건연산자

조건연산자는 if, else문을 함축시킨것으로서 한줄로서 처리하는 연산자이다. 형식은 다음과 같다.

```
식 ? 식1 : 식2 ;
```

식이 참이면 결과는 식1이 되고 거짓이면 결과는 식2가 된다.

- 할당연산자

어떤 값을 변수에 할당하는데 사용되는 연산자이다.

표 5-4. 할당연산자

연산자	의 미
a=b	변수 a에 b를 할당
a+=b	a와 b를 더하여 a에 결과를 할당
a-=b	a에서 b를 덜고 a에 결과를 할당
a*=b	a와 b를 곱하여 a에 결과를 할당
a/=b	a를 b로 나누어 a에 결과를 할당
a%=b	a를 b로 나누었을 때 생기는 나머지를 a에 할당

- 형변환연산자

이 연산자는 원하는 자료형으로 변환할 때 사용한다.

형식은 다음과 같다.

```
(자료형)식;
```

형변환연산자를 리용하면 자료형을 임의로 변환하여 출력할수 있으며 입력할수도 있다. 형변환형식은 암시적방법과 명시적방법으로 구분된다.

암시적방법은 콤파일러에 의해 자동으로 수행되며 변환될 자료형의 정보를 잃어버리지 않게 작은 크기를 가지는 자료형을 크기가 큰 자료형으로 변환하는것을 말한다.

명시적방법은 사용자가 형변환연산자를 리용하여 자료의 형을 변환하는것을 말한다.

5.2.3. 명령문

- 조건문

조건문이란 주어진 조건에 따라 서로 다른 프로그람을 실행시키는 명령문을 말한다. 여기에는 if, if~else, switch문이 있다.

형식은 다음과 같다.

```
•   If
    If (조건식)식;
•   If~else
    If(조건식)식1;
    Else식2;
•   Switch
    Switch (식){
        Case 상수식1:
            식1;
        Case 상수식2:
            식2;
            …
        Case 상수식n:
            식n;
        default:
            식0;
    }
```

-반복문

반복문은 프로그람의 어떤 부분을 주어진 조건을 만족하는동안 반복해서 실행하는 명령문이다. 여기에는 for, while, do~while문이 있다.

```
•    for문
for (초기식;조건식;증감식)식
또는
for (초기식;조건식;증감식){식}
•    while
while(조건식){
        식
    }
•    do while
do {
    식…
}while(조건식);
```

- 뛰여넘기명령문

뛰여넘기명령문은 프로그람의 실행도중 실행순서를 지정된 곳으로 바꾸려고 할 때 사용하는 명령문으로서 break,continue 등이 있다.

break문은 현재의 실행을 중지하고 현재 위치하고있는 블로크밖으로 조종을 옮기려고 할 때 사용한다.

continue문은 반복문내에서 반복이 시작되는 위치로 조종을 옮기려고 할 때 사용한다.

형식은 다음과 같다.

```
Break;
Continue;
```

5.2.4. 지시문

- include지시문

이 지시문은 JSP문서에 정적인 또는 동적인 다른 문서를 삽입할 때 리용한다.

형식은 다음과 같다.

```
<%@ include file= "파일의 주소" %>
```

이 지시문은 JSP문서가 해석되는 시점에서 포함되는데 JSP파일이 콤파일되여 실행될 때 포함된 문서도 역시 콤파일된다. 포함되는 문서는 HTML파일이나 Text파일도 되고 동적인 파일인 JSP파일도 된다.

HTML파일을 포함하는 경우(JSP파일도 해당됨) 포함되는 문서에 <html>,

</html> 또는 <body>, </body>가 존재하여서는 안된다.

HTML문서에는 오직 한개의 <html>, </html>와 <body>, </body>가 있어야 하는데 포함되는 문서에 이미 존재한다면 열람기는 이 문서를 정상적으로 읽을수 없다.

실례:

```
<html>
<head>
<title> 실례 </title>
<meta
http-equiv=' content-type' content=' text/html;charset=big5' >
</head>
<body>
   날자출력프로그람
  <p> </p>
  현재 날자와 시간은
  <%@ include file=' ex.jsp' %>이다.</body>
  </html>
```

한편 우에서 포함시킨 ex.jsp파일은 다음과 같다.

```
<%@ page contentType=' text/html;charset=big5' %>
<%=(new java.utilDate()).toLocaleString() %>
```

- Page지시문

Page지시문은 JSP페지의 전체적인 속성을 설정해주는 역할을 한다.

형식은 다음과 같다.

```
<%@ page [language= "java" ]
          [extends= "package.class" ]
          [import= "{package.class|package.*},…" ]
          [sesson= "true|false" ]
          [buffer= "none|8kb|sizekb" ]
          [autofluse= " true|false" ]
          [isThreadSafe= " true|false" ]
          [info= "text" ]
          [errorPage= "relativeURL" ]
         [contentType= "mimeType[;charset=characterset]" | " text/html;
charset=big5" ]
          [isErrorPage= " true|false" ]
%>
```

이 지시문은 여러가지 속성들을 설정하는데 리용한다. 구체적인 설명은 아래에서 진행한다.

language= "java"

이것은 스크립트에서 사용할 언어를 설정하는데 이 설정은 스크립트, 선언, 표현식 등에 적용된다. 이 속성값으로 허용되는것은 《java》만이다.

extends= "package.class"

JSP파일과 함께 콤파일하려는 상위클라스를 지정할 때 사용한다.

import= "{package.class|package.*},…"

Java에서는 사용자가 프로그람을 효률적이고 쉽게 작성할수 있도록 여러가지 클라스를 만들어놓고 그 클라스를 모아놓은 패키지를 제공한다. JSP에서도 이 Java의 패키지를 사용하여 프로그람을 작성할수 있는데 이 패키지를 사용하려면 이 속성을 사용하여 자기가 리용할 패키지를 밝혀주어야 한다.

JSP문서는 사용자의 편리를 위해 다음과 같은 패키지를 기본적으로 반입해주는데 이것들은 특별히 밝혀주지 않아도 된다.

그 패키지는 java.lang.*, javax.servlet.*, javax.servlet.jsp.*, javax.servlet.http.*들이다.

그러나 이 외의 패키지들을 사용하려면 반드시 이 지시문을 리용하여 해당 클라스를 추가해주어야 한다.

sesson= "true|false"

이 속성은 사용자가 웨브페지에 접속하여 사용자별정보를 사용하겠는가, 사용하지 않겠는가를 결정하는것이다.

buffer= "none|8kb|sizekb"

JSP에서 완충기는 JSP페지를 미리 실행시켜 그 결과값을 저장해놓기 위하여 사용한다. JSP에서 사용되는 완충기의 크기는 보통 Kbyte단위로 설정되는데 기정값은 8Kbyte로 설정된다. 만일 완충기를 사용하지 않으려면 속성값을 none으로 설정하는데 이렇게 되면 JSP페지의 실행결과를 림시적으로 저장할 완충기가 없으므로 실행한 결과를 순차적으로 웨브열람기로 전송하게 되여 화면에 내용이 천천히 보여지게 된다.

autoflush= " true|false"

이 속성은 JSP페지의 실행결과를 저장하는 완충기의 크기를 설정한 경우 JSP실행결과에 따라 완충기의 용량을 초과하면 자동적으로 처리하기 위한 속성이다. 이 속성의 값이 true일 때 완충기에 가득 차면 자동으로 그 결과를 열람기로 전송하지만 false일 때에는 완충기가 가득차게 되면 오유가 발생되여 JSP실행이 중단된다. 그러므로 다시 완충기의 공간을 충분히 설정해주어야 한다. 만일 완충기의 속성값을 none으로 설정하였다면 autoflush의 속성값은 오직 true만이 가능하다.

isThreadSafe= " true|false"

이 속성은 JSP페지에서 스레드실행을 허용할것인가를 설정하는것이다. 기정값은 true인데 그것은 JSP가 동시에 발생하는 의뢰기의 요청을 받아들여 다중으로 실행할수 있도록 하겠다는 의미이다. 만일 false인 경우 JSP페지는 동시에 스레드를 실행할수 없게 되여 단 한개의 의뢰기요구에만 응답할수 있다.

info= "text"

이 속성은 JSP페지에 대한 생략정보를 쓰려고 할 때 리용한다. 이 내용은 Servlet.getServletInfo()메쏘드를 리용하여 열람기에 출력할수 있다.

[errorPage= "relativeURL"]

이 속성은 오유가 발생하게 되면 해당 설정한 주소의 파일로 이동하여 그 페지가 출력되도록 하려고 할 때 리용한다.

isErrorPage= "true|false"

현재 실행되고있는 JSP페지에서 오유가 발생하는 경우 오유처리를 할수 있도록 할것인가를 설정하는 속성이다. 만일 이 속성이 true로 설정된다면 내장객체인 exception객체를 사용할수 있다.

contentType= "mimeType[;charset=characterset]" | " text/html;charset=big5"

JSP페지가 실행되여 그 결과를 열람기로 전송할 때 웨브열람기가 그 정보를 보여주는 방법을 나타내는 속성이다.

이 속성은 page지시문의 여러가지 속성들가운데서 반드시 설정해주어야 한다.

-Taglib지시문

JSP에서는 사용자정의형꼬리표를 사용할수 있다. 사용자정의형꼬리표를 사용하기 위해서는 사용할 사용자정의형꼬리표의 원천위치를 JSP페지에 알려주어야 한다. 이 역할을 하는것이 바로 Taglib지시문이다.

형식은 다음과 같다.

```
<%@ taglib url= "URLToTagLibrary"  prefix= "tagPrefix"%>
```

여기서 url은 사용자정의형꼬리표의 유일한 식별자를 지정하는 역할을 하는데 사용자정의형꼬리표가 있는 위치의 주소를 입력할수도 있고 사용자정의형꼬리표의 이름 또는 경로명을 입력할수도 있다. 그리고 prefix는 사용자정의형꼬리표의 접두어를 설정해주는것이다. 만일 접두어를 public로 설정하였다면 <public: loop></ public: loop>와 같이 사용자정의형꼬리표를 사용하여야 한다.

제3절. JSP의 내장객체들

5.3.1. request 객체

request객체는 동적인 웨브페지를 만드는데서 없어서는 안될 중요한 객체이다. request객체는 JSP페지에 접속한 사용자가 전달한 자료나 그외의 의뢰기정보, 페지정보 등을 저장하고 처리하는 객체이다. 사용자가 정보를 입력하고 정보를 전송하면 웨브페지는 정해진 방법으로 정보를 전송하고 그 정보를 받은 JSP페지는 request객체를 리용하여 그 전송된 정보를 분류하고 다시 사용할수 있는 정보로 변환한다.

JSP에 내장되여있는 Request객체는 javax.servlet.http.Http Servlet Request대면의 객체이다.

또한 이 대면은 javax.servlet.servletRequest대면을 상속받은것이므로 우의 대면들에 정의되여있는 메쏘드들을 사용할수 있다.

- getParameter메쏘드

이 메쏘드는 홈으로부터 전달된 값을 그 홈실체의 이름을 가지고 입력된 값을 알아내는 메쏘드이다. 이것은 요청파라메터에 할당된 값을 귀환한다. 지정된 이름의 파라메터가 존재하지 않는 경우 null을 귀환한다.

형식은 다음과 같다.

```
Java.lang.String getParameter(java.lang.String name)
```

이 메쏘드는 매개 변수로 문자렬자료형인 name을 필요로 한다. 여기서 name은 값을 전달하는 HTML의 input나 단추 등의 이름이다. 이 메쏘드는 이름에 해당하는 홈에 입력된 값을 문자렬자료형으로 반환한다.

- getParameterNames와 getParameterValues메쏘드

이 메쏘드들은 홈을 통해 사용자가 정보를 입력하고 입력한 자료를 전송하였을 때 그 자료의 이름을 일괄적으로 알아내고 그 내용을 가져오는데 리용한다.

이 메쏘드들의 형식은 다음과 같다.

```
    java.util.Enumeration getParameterNames(java.lang.
String[])
    getParameterValues(java.lang.string name)
```

getParameterNames는 전달받은 대화칸(form)들의 이름을 배렬객체로 반환하고 getParameterValues는 실체의 이름을 매개 변수로 전달받아 그 이름에 해당하는 실체의 값을 문자렬객체의 배렬로 반환한다.

- getServerName, getServerPort, GetRemoteAddr, getRemoteHost

이 메쏘드들은 Request객체를 리용하여 현재 JSP페지를 실행시키고있는 봉사기의 정보와 JSP페지를 요청하고있는 의뢰기의 정보를 알아내는데 리용된다.

5.3.2. response 객체

response객체는 완충기의 설정을 읽어오거나 새롭게 설정을 바꾸려고 할 때 리용한다. 즉 문자모임을 바꾸기, MIME형식을 변경하거나 완충기의 크기를 조절하는 기능을 수행한다.

JSP에 내장되여있는 response객체는 java.servlet.http.Http Servlet Response대면의 객체이며 또한 java.servlet.http.Http Servlet Response대면은 java.servlet.ServletResponse대면을 계승하여 생성된다. 그러므로 JSP의 Response객체는 java.servlet.ServletResponse, java. servlet.http.Http Servlet Response대면의 메쏘드를 모두 사용할수 있다.

- getBufferSize, setBufferSize메쏘드

완충기의 크기가 설정되면 JSP페지는 그 완충기를 사용하는데 JSP페지의 실행결과가 출력완충기의 크기를 초과하게 되면 자료넘침(Flush)이 발생한다. 이것은 단순히 완충기의 용량초과로 자료가 류실된다는것이 아니라 완충기에서 넘친 내용이 의뢰기로 전송된다는것이다. 이런 오유가 발생하게 되면 완충기를 사용하는 의미가 없어지기때문에 반드시 완충기의 크기를 조절하여야 한다.

이를 위해 getBufferSize와 setBufferSize메쏘드를 사용한다.

형식은 다음과 같다.

```
Response.getBufferSize()
Response.setBufferSize(1000000)
```

- getCharacterEncoding메쏘드

이 메쏘드는 문자변환방식을 바꾸거나 재설정하는 경우 사용한다.

실례:

```
<%= response.getCharacterEncoding() %>
```

- sendRedirect메쏘드

이 메쏘드는 페지를 강제로 이동시키는 기능을 수행한다. 이 메쏘드는 매개 변수로 이동시킬 페지의 위치정보를 입력받아 해당 페지를 의뢰기로 전송한다.

형식은 다음과 같다.

```
Response. sendRedirect(“이동시킬 주소”)
```

- sendError메쏘드

웨브봉사기와 웨브열람기는 거점에 접근할수 없는 경우와 봉사기가 접근을 거부하는 경우의 오유 등을 번호로 구분하여놓고 만일 접속중 그에 해당한 오유번호를 검출하게 되면 그 페지를 보여주게 된다. 이 메쏘드는 매개 변수로 오유번호를 넘겨받아 의뢰기에게 그 번호를 넘겨준다. 그러면 그 번호를 넘겨받은 의뢰기는 그 번호에 해당하는 페지를 출력한다.

실례:

response.sendError(401);

5.3.3. session 객체

session객체는 의뢰기의 대화접속정보를 관리하는 객체이다. session객체를 설정하는것은 봉사기에 접속하는 매개 의뢰기에 일종의 저장용완충기를 고유하게 할당하는것과 같다. 이 session객체가 할당될 때 먼저 Active session을 최대로 얼마나 생성할것인가를 설정하고 또한 사용자가 아무런 동작을 취하지 않고 대화접속을 유지할수 있는 최대제한시간을 설정하여 그에 맞게 session객체를 할당하게 된다.

대화접속의 식별부호(ID)가 봉사기에 의해 의뢰기에게 할당되는 방법은 란수를 리용하여 실현한다. 그러므로 의뢰기뿐아니라 봉사기역시 의뢰기에게 어떤 식별부호가 부여되겠는지 알수 없다.

- getId메쏘드

이 메쏘드는 대화접속의 식별부호를 얻는데 리용한다. 이 메쏘드는 의뢰기에게 할당된 대화접속의 식별부호값을 문자렬형으로 귀환한다.

실례:

```
<%= session.getId() %>
```

- getCreationTime()메쏘드

이 메쏘드는 대화접속이 생성된 시간을 알아내는 기능을 수행한다. 이것은 의뢰기가 접속하여 생성된 대화접속의 생성시간을 long형으로 귀환한다.

실례:

```
<%= session.getCreationTime() %>
```

- setAttribute, getAttribute메쏘드

처음 의뢰기에 할당된 대화접속에는 빈 완충기만 할당된다. 이 대화접속을 사용하려면 그 대화접속의 완충기에 의뢰기가 전송한 값을 속성값으로 할당하여야 한다. 매개 의뢰기가 JSP페지에 전송한 자료는 해당 페지에서만 그 값을 사용할수 있다.

즉 매개 요청자료는 그 페지내에서만 그 값을 유지할수 있으며 열람기를 리용하여 페지를 바꾸게 되면 그 값은 모두 완충기에서 없어진다. 그러나 필요에 따라서 그 값들을 유지하여야 할 때가 있는데 이때 페지에 상관없이 의뢰기가 완료될 때까지 그 속성들을 유지하자면 session객체에 그 값들을 기억시키면 된다.

이 기능을 수행하는것이 바로 이 메쏘드들이다.

형식은 다음과 같다.

```
setAttribute(java.lang.string name,java.lang.object value)
getAttribute(java.lang.string name)
```

setAttribute메쏘드는 현재 대화접속에 새로운 이름의 속성을 만들고 그 속성에 값을 할당해주며 getAttribute메쏘드는 대화접속에 저장되여있는 속성값들중 name에 해당하는 값을 java.lang.object의 객체형으로 귀환해준다.

실례:

```
<% session.setAttribute(“user”,sessionid) %>
<% session.getAttribute(“user”) %>
```

- getMaxInactiveInterval메쏘드,setMaxInactiveInterval메쏘드

사용자가 웨브열람기를 끄지 않았을 때와 일정한 시간이 지나도록 사용자가 아무런 일을 하지 않는 경우 대화접속을 해제시킬 필요가 있다.

기본적으로 JSP페지는 존재시간(SessionTimeout)이 설정되여있다. 이 시간을 알아내는것이 바로 getMaxInactiveInterval메쏘드이다.

이 메쏘드는 사용자가 대화접속을 실현하였을 때 아무것도 하지 않는 경우 자동으로 대화접속을 해제시키도록 시간을 설정하는데 리용한다. 이 메쏘드는 자동결속되는 시간을 초단위로 귀환한다. 기정값으로 1800s 즉 30min으로 설정되여있다.

만일 이 시간을 줄이거나 늘이는 경우 setMaxInactiveInterval메쏘드를 사용한다.

형식은 다음과 같다.

```
getMaxInactiveInterval()
setMaxInactiveInterval(int interval)
```

실례:

```
<% session. setMaxInactiveInterval(600) %>
<%= session. getMaxInactiveInterval() %>
```

5.3.4. 그 밖의 객체들

- application객체

application객체는 모든 페지가 다 공유할수 있는 자료를 담고있으며 자료에 대한 작업을 하는 객체이다. application내장객체는 javax.servlet. ServletContext대면의 객체이다. ServletContext는 웨브응용프로그람내에 있는 모든 Servlet들을 관리하며 정보를 공유할수 있게 도와주는 역할을 수행한다. 즉 application내장객체는 써브레트 및 웨브응용프로그람자체의 정보를 다룰수 있으며 자료를 저장해서 공유하여 여러곳에서 사용될수 있게 한다. 그리고 pageContext객체와 같이 페지에 대한 조종을 다른 페지로 아주 주거나(forward) 림시로 주는(include) 기능도 수행할수 있다.

- out객체

out객체는 의뢰기의 요청에 대한 응답전송을 담당하는 내장객체이다. out객체는 pageContext객체의 메쏘드를 사용하여 JSP내부에서 자동생성된다. 이 객체는 의뢰기로 자료를 전송하기 위하여 반드시 필요한 객체이기때문에 JSP에 미리 내장객체로 만들어두었다.

실례:

```
<%@ page contentType= "text/html;charset = euc-kr"%>
<%@ page  buffer ="1kb"%>
<h1>out객체실례</h1><hr>
<%
        out.println("out객체에 대한 실례이다. "+"<br>");
        out.println("완충기의              크기:           "             +
out.getBufferSize()+"Bytes<br>");
        out.println("남은 완충기의 크기 : "
+out.getRemaining()+"Bytes");
         out.flush();
%>
```

실례에서는 page지시문을 통해서 1kbyte로 완충기크기를 설정하고 out객체를 리용하여 자료를 기록할 때 기록된 자료의 크기가 1kbyte가 되면 자동으로 의뢰기로 전송된다. 그리고 다시 완충기가 채워지면 자료가 전송된다.

out객체에서 getBufferSize()메쏘드는 설정되여있는 완충기의 크기를 얻어내는 메쏘드이고 getRemaining메쏘드는 완충기의 남은 크기를 얻는 메쏘드이며 flush()메쏘드로는 완충기에 자료가 채워지면 자동으로 의뢰기로 반환되도록 하고있다. Out객체를 가장 많이 사용하는것은 println()메쏘드이다.

- config객체

config객체는 해당 페지에 대한 써브레트자료를 담고있으며 이 자료에 대한 작업을 하는 객체이다. config내장객체는 javax.sevlet.ServletConfig대면의 객체이다.

ServletConfig는 Servlet에 Servlet를 초기화하는동안 참조해야 할 정보를 전해주는 역할을 한다. 즉 Servlet가 초기화될 때 참조해야 할 다른 여러 정보를 가지고있다가 전해준다.

하지만 JSP는 페지를 요청할 때에 Servlet가 동적으로 생성되므로 config객체는 Servlet초기화의 메쏘드로는 거의 쓰지 않는다. 다른 방법으로 객체를 얻어 그것으로 웨브응용프로그람의 정보 등을 알아내는데 사용된다.

아래의 실례는 config내장객체를 리용해서 봉사기의 정보를 알아보는 실례이다.

실례:

```
<%@ page contentType="text/html; charset=euc-kr" %>
<html><body>
<h2> 내장객체 config를 사용한 정보출력 </h2>
<h3>
<%
out.println("Servlet이름 : " + config.getServletName()+"<br><br>");
ServletContext context = config.getServletContext();
out.println("봉사기판본번호 : "+context.getMajorVersion()+"."+context
.getMinorVersion());
%>
</h3><body><html>
```

우의 실례에서는 먼저 config객체를 사용하여 Servlet의 이름을 얻고 그 다음 config를 리용하여 ServletContext의 객체를 얻어서 봉사기의 정보를 출력한다. 하지만 여기서 얻어낸 ServletContext context는 application내장객체와 동일하기때문에 이러한 방법으로 ServletContext를 얻을 필요는 없다.

- page객체

page객체는 해당 페지자체를 참조하기 위한 객체로서 JSP페지에서 생성되는 Servlet 객체를 참조하는 참조변수이다. 즉 자기가 생성할 Servlet객체를 참조하는 객체이다. 그래서 자기 자신을 참조하는 this를 사용해서 생성한다. page객체는 내부적으로 자기자신을 참조하는 this를 받지만 형은 Object형이다.

실례:

```
<%@ page contentType= "text/html;charset = euc-kr"%>
<h1> page내장객체 </h1><hr>
<%! int sum(int a , int b){
        return a+b;
    }
```

```
%>
<%
    int aaa = 4, bbb =5;
out.println("처음 생성된 객체를 리용해서 출력:" +sum(aaa,bbb)+"<br>");
out.println("this 객체를 리용해서 출력 : " +this.sum(aaa,bbb)+"<br>");
    pageTest$jsp pagetest = (pageTest$jsp)page;
    out.println("page객체를 리용해서 출력 : "+pagetest.sum(aaa,bbb));
%>
```

- exception객체

exception객체는 해당 페지실행시 오유가 발생했을 때 처리하지 못한 레외정보와 이와 관련된 처리를 하는 객체이다. exception객체는 java. lang. Throwable 클라스의 JSP내장객체이다. 이 객체는 해당페지실행시 Servlet가 처리하지 못한 오유가 발생할 때 레외적으로 처리할 페지를 지정하였을 경우 지정된 페지로 레외적으로 전달하는 역할을 하는 객체이다.

제4절. JSP의 응용

5.4.1. 게시판에 글쓰기

게시물을 입력하자면 먼저 자료기지에 게시물입력을 위한 표(표이름:guest)가 창조되여있어야 한다.

창조한 표의 마당값들에 대하여 표 5-5에 주었다.

표 5-5. 게시물자료표(guest)의 구조

마당이름	자료형	크기	NULL	설 명
idx	int	4	no	자동증가,기본열쇠
no	int	4	no	게시물번호
subject	varchar	100	no	게시물제목
name	varchar	50	no	게시물이름
email	varchar	100	yes	게시자전자우편주소
content	text	16	no	게시물내용
indate	datetime	8	no	게시날자
pwd	varchar	10	no	게시물암호

게시판에 글을 쓰기 위하여 웨브페지를 두개의 페지로 구분한다. 하나는 입력을 위한 페지이고 다른 하나는 입력을 처리하기 위한 페지이다.

-게시물의 내용을 입력하는 페지(write.jsp)

```
<html>
<head>
<title>게시판글쓰기</title>
<style type= “text/css” >
<!—
.normal{font-family:
“ WKLChongbong” ;font-size:x-small;font-style:normal;font-weight:normal;text-decoration:none}
.normalbold{font-family:
“WKLChongbong” ;font-size:x-small;font-style:normal;font-weight:bold;text-decoration:none }
-->
```

```
</style>
<script language= "javascript" >
function checkinput(){
 //제목이 입력되였는가를 확인한다.
 if (document.guest.subject.value== "" ){
 alert( "제목을 입력하시오." );
 return;
}
 //이름이 입력되였는가를 확인한다.
 if (document.guest.name.value== "" ){
 alert( "이름을 입력하시오." );
 return;
}
 //암호가 입력되였는가를 확인한다.
 if (document.guest.pwd.value== "" ){
 alert( "암호를 입력하시오." );
 return;
}
 //내용이 입력되였는가를 확인한다.
 if (document.guest.content.value== "" ){
 alert( "내용을 입력하시오." );
 return;
}
//사용자의 입력을 전달한다.
document.guest.submit();
}
</script>
</head>
<body bgcolor= "#ffffff" >
<form method= "post" action= "writeproc.jsp" name= "guest" >
 <table width= "500" border= "0" cellspacing= "0" cellpadding= "0" >
    <tr>
        <td colspan= "2" height= "35" >
        <div align= "center" ><font size= "5" >게시판</font></div>
```

```
        </td>
      </tr>
      <tr>
        <td colspan= “2” class= “normal” height= “42” >
          여러분들의 글을 남겨주시오.
        </td>
      </tr>
</table>
<table width= “ 500” border= “ 1” cellspacing= “ 0” cellpadding= “ 0”
bordercolor= “#000000” >
      <tr>
        <td width= “211” class= “normalbold” bgcolor= “#ccccff” >
          <div align= “center” >게시물제목</div>
        </td>
        <td width= “370” >
          <input type= “text” name= “subject” size= “50”
                      maxlength= “100” >
        </td>
      </tr>
      <tr>
        <td width= “211” class= “normalbold” bgcolor= “#ccccff” >
          <div align= “center” >게시자이름</div>
        </td>
        <td width= “370” >
          <input type= “text” name= “name” size= “50”
                     maxlength= “100” >
        </td>
      </tr>
      <tr>
        <td width= “211” class= “normalbold” bgcolor= “#ccccff” >
          <div align= “center” >게시자전자우편주소</div>
        </td>
        <td width= “370” >
          <input type= “text” name= “email” size= “50”
```

```
                    maxlength= “100” >
        </td>
    </tr>
    <tr>
        <td width= “211” class= “normalbold” bgcolor= “#ccccff” >
            <div align= “center” >암호</div>
        </td>
        <td width= “370” >
            <input type= “password” name= “pwd” size= “50”
                    maxlength= “100” >
        </td>
    </tr>
    <tr bgcolor= “#ccccff” >
        <td colspan= “2”  >
            <div align= “center” class= “noramlbold” >게시물내용</div>
        </td>
    </tr>
    <tr>
        <td colspan= “2” >
            <textarea name= “content” rows= “20” cols= “70” >
            </textarea>
        </td>
    </tr>
    <tr>
        <td colspan= “2” >
            <div align= “center” >
            <input type= “button” name= “insert” value= “등록”
                    onclick= “checkinput()” >
            <input type= “reset” name= “reset”  value= “취소” >
        </div>
        </td>
    </tr>
</table>
</form>
```

```
    </body>
    </html>
```

-게시물입력처리를 위한 페지(writeproc.jsp)

```
<%@ page contentType= “text/html;charset=euc-kr”
     import= “java.sql.*,java.util.*”
     errorPage= “error.jsp”
%>
<html>
<head>
<title>게시판 글쓰기</title>
<style type= “text/css” >
<!—
.normal{font-family:
 “WKLChongbong” ;font-size:x-small;font-style:normal;font-weight:normal;
text-decoration:none}
.normalbold{font-family:
 “ WKLChongbong” ;font-size:x-small;font-style:normal;font-weight:bold;te
xt-decoration:none }
-->
</style>
</head>
<%
try{
     //사용자가 입력한 자료를 저장한다.
     String subject=request.getParameter( “subject” ),
          name=request.getParameter( “name” ),
          email=request.getParameter( “email” ),
          content=request.getParameter( “content” ),
          pwd=request.getParameter( “pwd” );
     //클라스를 적재한다.
     Class.forName( “sun.jdbc.odbc.jdbcOdbcDriver” )
     //련결명령어를 작성한다.
     String url= “jdbc:odbc:JspDB” ;
     //자료기지를 련결한다.
```

```
    Connection insCon=DriverManager.getConnection(url, "sa" ,
                                                    "dhslove" );
    //질문을 실행할 PreparedStatement객체를 생성한다.
    PreparedStatement pstmt1=null,pstmt2=null;
    //no의 값을 설정하기 위하여 질문을 작성한다.
    String jsql= "select max(no) from guest" ;
    //Connection객체의 PreparedStatement메쏘드를 리용하여
  PreparedStatement객체에 질문을 할당한다.
    pstmt1=insCon.prepareStatement(jsql);
    //no의 값을 설정하기 위하여 질문을 실행한다.
    ResultSet rs=pstmt1.executeQuery();
    //no의 값을 설정한다.
    int no=1;
    If (!rs.wasNull()){
        Rs.next();
        No=rs.getInt(1)+1;
    }
  //객체를 닫는다.
  Rs.close();
  //indate의 값을 설정한다.
  Calender dateIn=Calender.getInstence();
  String indate=Integer.toString(dateIn.get(Calender.YEAR))+ "-" ;
    indate=indate+ Integer.toString(dateIn.get(Calender.MONTH))+ "-" ;
    indate=indate+ Integer.toString(dateIn.get(Calender.DATE))+ "-" ;
    indate=indate+ Integer.toString(dateIn.get(Calender.HOUR_OF_DAY))+
     ":" ;
    indate=indate+ Integer.toString(dateIn.get(Calender.MINUTE))+ ":" ;
    indate=indate+ Integer.toString(dateIn.get(Calender.SECOND))+ "-" ;
//질문을 작성한다.
    jsql= " INSERT INTO guest (no,subject,name,email,
    content,indate,pwd)" ;
jsql=jsql+ "VALUES(?,?,?,?,?,?,?)" ;
pstmt2=insCon.prepareStatement(jsql);
//질문의 ?부분을 완성한다.
```

```
pstmt2.setInt(1,no);
pstmt2.setString(2,subject);
pstmt2.setString(3,name);
pstmt2.setString(4,email);
pstmt2.setString(5,content);
pstmt2.setString(6,indate);
pstmt2.setString(7,pwd);
//질문을 실행한다.
Pstmt2.executeUpdate();
Response.sendRedirect(list.jsp);
%>
<body bgcolor= “#ffffff” >
<%
}
catch (SQLException e){
%>
<table width= “270” border= “0” cellspacing= “0” cellpadding= “5” >
    <tr bgcolor= “#3399cc” >
            <td  height=  “  39” class=  “  normalbold” >게시판에  게시되지
            않았다.</td>
        </tr>
        <tr>
            <td class= “normal” >
        <p>게시판에 문제가 있다.<br>관리자에게 문의해보시오.</p>
        <p align= “center” ><a href= “list.jsp” >목록</a></p>
        </td>
        </tr>
    </table>
    <%
}
%>
</body>
</html>
```

5.4.2. 게시판의 게시물목록 보기(list.jsp)

게시판을 시작하면 처음으로 목록이 현시된다. 현시되는 목록은 다음의 list.jsp파일에 의하여 실행된다.

```
<%@ page contentType= "text/html;charset=utf-8"
    import= "java.sql.*,java.util.*"
    errorPage= "error.jsp"
%>
<html>
<head>
<title>게시판 목록보기</title>
<style type= "text/css" >
<!—
.normal{font-family:
 "        WKLChongbong" ;font-size:x-small;font-style:normal;font-weight:
normal;text-decoration:none}
.normalbold{font-family:
 " WKLChongbong" ;font-size:x-small;font-style:normal;font-weight:bold;te
xt-decoration:none}
-->
</style>
</head>
<%
//목록의 idx를 보관한다.
String idx;
int listidx;
try{
    idx=request.getParameter( "idx" );
    listidx=integer.parseInt(idx);
}catch(Exception e){
    listidx=0;
}
Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
//련결명령문을 작성한다.
String url= "jdbc:odbc:JspDB" ;
```

```
//자료기지를 련결한다.
Connection listCon= DriverManager. getConnection(url,“sa”,“dhslove”);
//질문을 실행할 PreparedStatement객체를 생성한다.
PreparedStatement pstmt1=null, pstmt2=null;
//목록의 idx를 구하기 위해 표에 저장되여있는 자료의 개수를 구한다.
String jsql= “SELECT COUNT(*) FROM guest”;
pstmt1=listCon.PreparedStatement(jsql);
ResultSet cntRs=pstmt1.executeQuery();
cntRs.next();
int cnt=cntRs.getInt(1);
//10개 단위로 목록을 구분하여 목록의 개수를 가져온다.
int cntList=cnt/10;
int remain=cnt%10;
//목록을 가져오기 위해 질문을 작성한다.
Jsql= “SELECT * FROM guest ORDER BY no DESC”;
Pstmt2=listCon.PreparedStatement(jsql);
ResultSet rs=pstmt2.executeQuery();
%>
<body bgcolor= “#ffffff”>
<p><span class= “normalbold”>게시물목록</span></p>
<table width= “500” border= “1” cellspacing= “0” cellpadding= “2”>
<tr class= “normalbold”  bgcolor= “#ccccff”>
    <td class= “normalbold” width= “38”>
        <div align= “center”>번호<div>
    </td>
    <td class= “normalbold” width= “63”>
        <div align= “center”>게시자이름<div>
    </td>
    <td class= “normalbold” width= “81”>
        <div align= “center”>날자<div>
    </td>
    <td class= “normalbold” width= “249”>
        <div align= “center”>제목<div>
    </td>
```

```
</tr>
<%
//목록이 있으면 자료를 출력한다.
If (!rs.wasNull()){
  For(int i=0;i<listidx*10;i++)rs.next();
  Int cursor=0;
  Whill(re.next()){
    Int idx=re.getString( “idx” ),
    no=re.getInt(no);
    String subject=rs. getString( “subject” ),
        name= getString( “name” ),
        email= getString( “email” ),
        indate= getString( “indate” ),
%>
<tr>
    <td class= “normal” width= “38” >
        <div align= “center” ><%=no%><div>
    </td>
    <td class= “normal” width= “63” >
        <div align= “center” ><%=name%><div>
    </td>
    <td class= “normal” width= “81” >
        <div align= “center” ><%=indate.substring(0,10)%><div>
    </td>
    <td class= “normalbold” width= “249” >
        <div align= “ center” ><a href= “ view.jsp?idx=<%= idx %>” >
        <%=subject%><div>
    </td>
</tr>
<%
  cursor ++;
  if(cursor>=10) break;
}
}
```

```
%>
</table>
<br>
<table width= "500" border= "1" cellspacing= "0" cellpadding= "2" >
<tr>
    <td class= "normal" ><div align= "center" >
<%
if (listidx>0){
%>
<a href= "list.jsp?lidx=<%= listidx-1%>" >이전</a>
<%
}
if (listidx<cntList-1 || (listidx==cntList-1 && remain>0)){
%>
    <a href= "list.jsp?lidx=<%=listidx+1%>" >다음</a>
<%
}
%>
    <a herf= "write.jsp" >쓰기</a></div>
</td></tr></table>
</body>
</html>
```

5.4.3. 게시물의 내용보기(view.jsp)

게시판목록에서 해당 게시물을 선택하면 게시물에 대한 구체적인 내용을 볼수 있도록 하는 파일이다.

```
<%@ page contentType= "text/html;charset=utf-8"
    import= "java.sql.*,java.util.*"
    errorPage= "error.jsp"
%>
<html>
<head>
<title>게시판 내용보기</title>
<style type= "text/css" >
<!—
```

```
.normal{font-family:
 "        WKLChongbong";font-size:x-small;font-style:normal;font-weight:
normal;text-decoration:none}
.normalbold{font-family:
 " WKLChongbong";font-size:x-small;font-style:normal;font-weight:bold;te
xt-decoration:none }
-->
</style>
</head>
<%
    //사용자가 선택한 게시물의 번호를 얻는다.
    String idx=request.getParameter( "idx" );
    //자료기지련결을 위하여 클라스를 적재한다.
    Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
    //련결명령문을 작성한다.
    String url= "jdbc:odbc:JspDB" ;
    //자료기지에 련결한다.
    Connection   viewCon=   DriverManager.getConnection(url,    "  sa" ,
 "dhslove" );
    //질문을 작성한다.
    String jsql= "SELECT * FROM guest WHERE idx=?" ;
    //질문을 실행할 PreparedStatement객체를 생성한다.
    PreparedStatement pstmt=null;
    pstmt=viewCon.preparedStatement(jsql);
    //?로 되여있는 부분을 setString을 리용하여 완성한다.
    Pstmt.setInt(1,Integer,parseInt(idx));
    //질문을 실행하여 ResultSet객체에 결과를 저장한다.
    ResultSet rs=pstmt.executeQuery();
%>
    <body bgcolor= "#ffffff" >
<%
//자료를 가져왔다면 값을 출력한다.
If (!rs.wasNull()) {
    Rs.next();
```

```
    String subject=rs.getString( “subject” );
    name=rs.getString( “name” );
    email=rs.getString( “email” );
    content=rs.getString( “content” );
    indate=rs.getString( “indate” );
    pwd=rs.getString( “pwd” );
%>
<table width= “500” cellpadding= “0” cellspacing= “0” border= “0” >
    <tr>
        <td colspan= “2” height= “35” >
        <div align= “center” class= “normalbold” >게시물내용보기</div>
        </td>
    </tr>
</table>
<table width= “500” cellpadding= “0” cellspacing= “0” border= “1”
            bordercolor= “#000000” >
    <tr>
        <td width= “211” bgcolor= “#ccccff” >
        <div align= “center” class= “normalbold” >제목</div>
        </td>
        <td width= “370” class= “normal” ><%= subject%></td>
    </tr>
    <tr>
        <td width= “211” bgcolor= “#ccccff” >
        <div align= “center” class= “normalbold” >게시자이름</div>
        </td>
        <td width= “370” class= “normal” ><%= name%></td>
    </tr>
    <tr>
        <td width= “211” bgcolor= “#ccccff” >
            <div    align=    “    center” class=    “    normalbold” >
            게시자전자우편주소</div>
        </td>
        <td width= “370” class= “normal” ><a href= “mailto:<%= email%>” ><%=
```

```
        email%></td>
    </tr>
    <tr>
        <td colspan= “2” width= “211” bgcolor= “#ccccff” >
        <div align= “center” class= “normalbold” >내용</div>
        </td>
    </tr>
    <tr>
        <td colspan= “2”  class= “normalbold” >
          <textarea name= “content” rows= “20” class= “normal” >
              <%=content%>
          </textarea>
        </td>
    </tr>
</table>
<%
//ResultSet를 닫는다.
Rs.close();
}
%>
<table width= “500” border= “0” cellpadding= “0” cellspacing= “0” >
    <tr><td colspan= “2” height= “35” >
        <div align= “center” class= “normalbold” >목록보기</div>
        </td>
    </tr>
</table>
</body>
</html>
```

제 6 장. PHP 언어

제1절. PHP언어의 기초

6.1.1. PHP 의 개념

PHP는 HTML문서내부에 포함되여 웨브봉사기에서 실행할수 있는 스크립트언어이다.

실례:

```
<html><head><title>실례</title></head>
<body>
<?php echo " 이것이 PHP스크립트입니다!"; ?>
</body></html>
```

Perl이나 C언어와는 달리 HTML을 출력하는데는 많은 명령어가 필요없다. HTML안에 하려고 하는 내용에 대한 스크립트코드를 써주면 된다.

PHP코드는 <?php 와 ?>괴리표사이에 입력하여야 한다.

PHP가 의뢰기측 Javascript와 다른 점은 이 코드가 봉사기에서 실행된다는것이다.

PHP로 무엇을 할수 있는가를 고찰해보자.

기본적으로 PHP는 CGI프로그람에서 할수 있는 모든것을 할수 있다. 즉 홈페지의 자료를 가져오고 동적인 웨브페지를 만들거나 쿠키를 보내고 받을수도 있다.

HTML문서들은 문서작성자의 의도에 따라 항상 고정된 정보를 보여주었다. 즉 그림을 어디에 어떻게 띄우라고 명령을 주고 글자는 어디에 넣으라는 명령을 주게 되면 문서는 항상 작성자가 지시한 화면만을 보여주었다. 이러한 문서를 정적문서라고 한다. 그러나 작성자를 포함한 방문객이나 다른 사람들로부터 정보를 입력받아서 적절하게 내용이 작성되는 문서를 동적문서라고 한다. 쉽게 말해서 관리자가 아니더라도 사용자가 관리자의 홈페지에 변화를 줄수 있는 문서라는것이다.

CGI는 이러한 동적문서를 작성하는 수단을 제공한다. CGI를 실현한 대표적인 레가 방문록, 게시판, 실시간대화 등을 들수 있다. 여기서 사용자는 방문록에 글을 남김으로써 봉사기의 방문록문서에 변화를 주게 된다.

CGI의 목적은 우에서 본 실례와 같이 HTML만으로 실현할수 없는것을 실현하자는데 있다. CGI의 원리를 보면 사용자가 열람기를 통하여 봉사기에 원하는 정보를 보내면 홈페지의 봉사기는 사용자가 요청한 CGI프로그람을 동작시켜 그 결과를 사용자에게 되돌려주는것이다. 즉 호상작용이 일어나는것이다. 이와 같이 CGI를 사용하면 단순히 보여주는 웨브페지가 아니라 방문자와 정보를 주고받을수 있는 웨브페지를 구축할수 있다. 이것이 CGI의 기본개념이다.

PHP로는 자료기지와의 련동을 진행할수 있다.

PHP는 다음과 같은 자료기지봉사기들을 지원해준다.

Adabas D	InterBase	Solid
dBase	mSQL	Sybase
Empress	MySQL	Velocis
FilePro	Oracle	Unix dbm
Informix	PostgreSQL	

PHP는 IMAP나 SNMP, NNTP, POP3, HTTP 등의 통신규약들을 사용하여 다른 봉사기들에 접근하며 자료를 교환할수 있다. raw network소케트를 사용하면 그외의 통신규약들을 사용할수도 있다.

PHP에서는 대소문자를 구분하여 작성하며 함수이름들은 대소문자를 구분하지 않아도 된다. 그러나 편의상 모든 함수이름이나 변수이름들은 대소문자를 구분해주는것이 좋다. PHP코드는 HTML과 구분하기 위하여 《<?》와 《?>》를 리용하며 매 명령문들은 반두점(;)으로 구분한다.

실례:

```
<html>
  <body>
  <font size=2 color=blue>이것은 HTML문서에서
        출력되는 내용이다.</font>
  <br>
  <?
  echo ("<font size=2 color=red>이것은 PHP코드에서 출력되는
        내용이다.</font>");
 ?>
 </body>
</html>
```

PHP에서 해설문은 여러가지 방법으로 표현한다.

- 해설문이 여러 행인 경우

/* . . . */

- 해설문이 한 행인 경우

//

#

실례:

```
<html>
```

```
<body>
   <!-- 이것은 html해설문입니다. -->
   <font size=2 color=blue>이것은 HTML문서에
       해당하는 내용이다.</font>
   <br>
   <?php
   /* 이것은
   php
   */
   echo ("<font size=2 color=red>이것은 PHP코드에
        해당하는 내용이다.</font>");
   // 이것도 php해설문이다.
   # 이것도 php해설문이다.
   ?>
</body>
</html>
```

6.1.2. PHP의 력사

PHP는 1994년 가을 Rasmus Lerdorf가 처음 만들었다. 처음에 비공개로 그의 홈페지에 사용되다가 1995년초부터 외부에 공개되여 Personal Home Page Tools라고 불리우게 되였다. 이것은 몇개의 특별한 마크로를 사용할수 있는 단순한 구문해석기(parser)엔진과 게시판이나 계수기같이 홈페지의 뒤에서 공통적으로 사용할수 있는 몇개의 간단한 편의프로그람으로 구성되였다.

이 구문해석기가 1995년 중엽에 재작성되여 PHP/FI Version 2라고 명명되였다. FI는 Rasmus가 작성한 HTML형식의 자료를 해석할수 있는 별도의 묶음이다. 그는 이 두가지를 합치고 mSQL을 지원하기 위하여 PHP/FI를 내놓았다. PHP/FI는 빠른 속도로 발전하였고 많은 사람들이 여기에 공헌을 하였다. 정확한 통계는 없지만 1996년 후반기에 PHP/FI는 전세계적으로 최소한 15,000개이상의 웨브페지에서 사용되는것으로 추정되였고 1997년 중반기에는 그 수가 50,000으로 늘어났다.

1997년 중엽 PHP는 또 다른 변화를 가져왔다. 이때부터 PHP는 Rasmus의 개인이 아닌 집단에 의해 개발되여왔다.

새 구문해석기를 Zeev Suraski와 Andi Gutmans가 재작성하였고 PHP Version 3이라는 이름으로 불렀다. PHP/FI의 기능들이 많이 옮겨졌고 그외에도 많은 기능들이 새로 작성되였다.

현재 PHP/FI나 PHP3은 C2의 StrongHold web server나 RedHat Linux와 같은 여러 상업적인 제품과 함께 제공되고있으며 전 세계적으로 최소한 150,000개의 웨브거점에서 사용되고있다. 이 수는 인터네트에서 Netscape's flagship Enterprise server의 사용자수보다 많다.

새로운 PHP는 Zend와 같은 효률적인 스크립트엔진과 Apache이외의 웨브봉사기에서도 모듈로 실행될수 있도록 제작되고있다.

6.1.3. 기본문법

- HTML로부터 탈퇴

HTML로부터 탈퇴하여 《PHP코드방식》으로 들어가는 방법에는 4가지가 있다

실례:

```
1.<?
   echo ("이것은 가장 단순한 SGML처리명령이다.
     \n");
  ?>
2.<?php
   echo("만일 XML문서들을 봉사해주려고 한다면 이와 같이 하시오 .
\n"); ?>
3.<script language="php">
   echo ("일부 편집기들은 FrontPage와 같이
                    처리명령들처럼 하지 못한다.");
 </script>
4.<%
  echo ("마음대로 ASP격식끄리표들을 사용할수 있다."); %>
 <%= $variable; # 이것은 "<%echo .."를 위한 지름길이다 %>
```

첫번째방법은 short tag가 설정되여있어야 사용가능하다. php.ini파일에서 short tags=On으로 설정하거나 콤파일할 때 "—enable-short-tags"선택항목을 주어야 한다.

네번째방법은 ASP-style tags가 설정되여있어야 사용가능하다. php.ini파일에서 asp_tags =On으로 설정하거나 콤파일할 때 "--enable-asp_tags"선택항목을 주어야 한다. 여기서 ASP-style tags는 3.0.4부터 지원된다.

- 명령구분(Instruction separation)

매개 명령은 C나 Perl과 마찬가지로 반두점(;)으로 구분한다.

PHP의 완료끄리표인 "?>"는 문장의 끝이라는 의미도 가지고있다. 다음 두개의 문장은 동일하다.

실례:

```
<?php
      echo "이것이 하나의 시험이다.";
?>
<?php echo "이것이 하나의 시험이다." ?>
```

- 해설문(Comments)

PHP는 C와 C++, Unix shell형태의 해설문을 제공한다.

실례:

```
<?php
    echo "이것이 하나의 시험이다."; //이것은 C++격식해설문이다.
      /* 이것은 여러행 해설문이다.
         역시 또 다른 행 해설문 */
     echo "이것은 여전히 또 다른 시험이다.";
     echo "최종시험"; # 이것은 shell격식해설문이다.
?>
```

"one-line"해설문은 행의 끝이나 현재 php코드블로크의 끝에서 끝난다.

```
<h1>이것은 하나의 <?# echo "simple";?> 실례이다.</h1>
```

C형태의 해설문은 중복해서 사용하지 말아야 한다.

제2절. 변수와 상수

6.2.1. 변수형

PHP는 변수형 즉 배렬형, 실수형, 옹근수형, 객체형, 문자렬형을 지원한다.

변수형은 보통 프로그람작성자가 선택하지 않아도 된다. 그것은 PHP가 실행중에 변수값의 내용에 따라 자동적으로 바꾸어주기때문이다.

만약 특정한 변수형으로 지정하고싶다면 변수를 변환하거나 settype()함수를 사용한다.

종종 변수들은 실행할 때 어떤 형으로 되여있는가에 따라 특정한 상황에서 원하는대로 동작하지 않을 때가 있다.

- 옹근수형은 다음과 같은 형식으로 표현한다.

  $a = 1234; # 10진수

  $a = -123; # 부수

  $a = 0123; # 8진수

  $a = 0x12; # 16진수

- 실수형은 다음과 같은 형식으로 표현한다.

  $a = 1.234; $a = 1.2e3;

- 문자렬은 두개의 구분기호를 사용하여 나타낸다.

  문자렬을 2중인용부호(")로 둘러싸서 표시하면 다음과 같이 특수문자들을 포함할수 있다. C나 Perl에서처럼 역사선(\)를 사용하여 특수기호를 표시한다.

표 6-1. 특수기호

기호	의미
\n	새로운 행
\r	자료전송
\t	태브
\\	역사선기호출력
\$	딸라기호출력
\"	2중인용부호출력

문자렬을 표현할 때 2중인용부호(")대신 단일인용부호(')를 사용할수도 있다. 이때에는 \\과 \'의 두개 특수기호만이 역사선처리되고 나머지 특수기호들은 그대로 출력된다.

단일인용부호를 사용하는 문자렬에서는 변수를 사용하여 그 값을 출력하는 기능을 사용할수 없다.

문자렬을 선언하는 다른 방법은 here doc구문("<<<")을 사용하는 방법이 있다. <<<뒤에 적당한 식별자를 적어주고 원하는 문자렬의 내용을 써넣은 후에 앞에 적은 식별자로 문자렬을 끝맺으면 된다. 끝맺는 식별자는 해당 행의 첫번째 칸에서 시작해야 한다.

실례:

```
$str = <<<EOD
    문자렬의 실례
    여러행 메꾸기(spanning)
    here doc문장 리용하기.
EOD;
```

다른 언어와 달리 PHP는 문자렬련결에 "+"연산자를 리용하는것이 아니라 "."를 리용한다.

문자렬안의 매개 문자에는 C언어에서의 문자렬배렬에서처럼 첨수(index)를 통해 접근한다.

실례:

```
<?php
/* 하나의 문자렬대입 */
$str = "이것이 하나의 문자렬이다.";
/* 그것에 덧붙히기 */
$str = $str . " 이 문자렬은 첨부된 문자렬이다.";
/* 행바꾸기*/
$str .= " 끝에서 행바꾸기가 진행된다.\n";
/* 이 변수는 옹근수 9이다. */
$num = 9;
/* 문자렬의 첫번째 문자를 가져온다.*/
$str = 'This is a test.';
$first = $str[0];
/* 문자렬의 마지막문자를 가져온다. */
$str = 'This is still a test.';
$last = $str[strlen($str)-1];
?>
```

문자렬변환에서 문자렬을 수자로 다룰 때 결과값과 형은 다음과 같이 결정된다.

만약 문자렬안에 《.》이나 《e》, 《E》의 문자가 있을 경우 그 형은 배정확도형(double)으로 된다. 그렇지 않으면 옹근수값이다.

문자렬이 정확한 수자자료로 시작되지 않으면 그 값은 0이다. 정확한 수자자료는 《+》, 《/》, 《-》, 《0》-《9》, 《.》과 수자뒤의 《e》나 《E》라는 표시가 붙는다.

첫번째 표현식이 문자렬형(string)인 경우 변수의 형은 첫번째가 아닌 두번째 표현식에 의해 결정된다.

실례:

```
$foo = 1 + "10.5";              // $foo는 실수 (11.5)
$foo = 1 + "-1.3e3";            // $foo는 실수(-1299)
$foo = 1 + "bob-1.3e3";         // $foo는 옹근수(1)
$foo = 1 + "bob3";              // $foo는 옹근수 (1)
$foo = "10.0 " + 1;        // $foo는 옹근수 (11)
$foo = "10.0 " + 1.0;      // $foo는 실수 (11)
```

만약 우의 실례결과를 실지로 확인하려면 우의 코드를 아래의 식에 삽입한다.

```
echo "\$foo==$foo; type is " .gettype( $foo ) . "<br>\n";
```

- 배렬형은 련상배렬(associative arrays)과 벡토르배렬(vectors)의 두가지로 동시에 사용된다.

· 1차원배렬

PHP는 스칼라배렬과 련상배렬을 지원한다. 실지로는 두 배렬의 차이가 없다. list()나 array()함수를 사용하여 배렬을 만들거나 매 원소의 값을 정해주어 배렬을 만들수 있다.

실례:

```
$a[0] = "abc";
$a[1] = "def";
$b["foo"] = 13;
```

또한 다음과 같이 변수에 값을 대입하는것만으로 배렬을 만들수도 있다.

실례:

```
$a[] = "hello"; // $a[2] == "hello"
$a[] = "world"; // $a[3] == "world"
```

배렬은 원하는 형식에 따라 asort(), arsort(), ksort(), rsort(), sort(), uasort(), usort(), uksort() 함수들을 리용해 순서대로 정렬할수 있다.

count()함수를 사용하면 배렬의 원소개수를 얻을수 있다.

next()와 prev()함수를 리용하여 배렬의 내용을 탐색할수 있다. 또한 each()함수를 사용할수도 있다.

· 다차원배렬(Multi-Dimensional Arrays)

다차원배렬이라 하더라도 실제로 매우 간단한다. 배렬의 매 차원에 대하여 단지 [key]값을 뒤에 붙여주면 된다.

실례:

```
$a[1] = $f;          // 1차원
$a["foo"] = $f;
$a[1][0] = $f;       // 2차원
$a["foo"][2] = $f;//(수값과 련상첨수들을 혼합할수 있다.)
$a[3]["bar"] = $f;// (수값과 련상첨수들을 혼합할수 있다.)
$a["foo"][4]["bar"][0] = $f;    // 4차원
```

PHP3이나 문자렬내에서 다차원배렬의 값에 직접 접근하는것은 불가능하다.

실례:

```
$a[3]['bar'] = 'Bob';
echo "This won't work: $a[3][bar]";
```

PHP3에서 우의 실례의 출력은 《This won't work: Array[bar]》이다. 원하는 결과를 얻으려면 문자렬련결연산자(.)를 사용하여야 한다.

실례:

```
$a[3]['bar'] = 'Bob';
echo "This will work: " . $a[3][bar];
```

PHP4에서는 배렬식을 중괄호({})안에 넣는다.

실례:

```
$a[3]['bar'] = 'Bob';
echo "This will work:{$a[3][bar]}";
```

여러가지 방법으로 다차원배렬에 값을 넣을수 있으나 련상(associative)배렬에 값을 저장하는 간단한 방법은 array()명령을 사용하는 방법이다.

다음의 두가지 코드는 1차원배렬에 값을 저장하는 방법과 동일하다.

실례 1:

```
$a["color"]= "red";
$a["taste"]= "sweet";
$a["shape"]= "round";
$a["이름"]= "apple";
$a[3]= 4;
```

실례 2:

```
$a = array(
    "color" => "red",
```

```
                "taste" => "sweet",
                "shape" => "round",
                "이름"  => "apple",
                3 => 4
            );
```

array()함수를 리용하여 다차원배렬을 표시하자면 다음과 같이 한다.

실례:

```
<?
 $a = array(
      "apple"   => array(
           "color"   => "red",
           "taste"   => "sweet",
           "shape"   => "round"
    ),
      "orange"   => array(
           "color"   => "orange",
          "taste"   => "tart",
          "shape"   => "round"
     ),
     "banana"   => array(
          "color"   => "yellow",
          "taste"   => "pastey",
          "shape"   => "banana-shaped"
     )
 );

 echo $a["apple"]["taste"];    # "sweet"를 출력한다.
 ?>
```

객체부분에서는 객체초기화를 진행하여야 한다.

객체(Object)의 초기화는 new명령을 사용하여 객체를 변수에 대입시키는것으로 실현한다.

실례:

```
class foo {
    function do_foo () {
```

```
                echo "Doing foo.";
            }
        }
        $bar = new foo;
        $bar->do_foo();
```

6.2.2. 형전환(type juggling)

PHP는 변수선언에서 명확한 정의를 할 필요도 없고 그것을 지원하지도 않는다. 변수의 형은 변수가 사용되는 코드속에서 결정된다. 즉 var라는 변수에 문자렬값을 할당하면 var는 문자렬변수가 되고 옹근수값을 할당하면 옹근수형변수가 된다.

PHP의 자동형전환의 실례는 《+》연산에서도 찾을수 있다. 어떤 하나의 연산수가 배정확도형이면 나머지 모든 연산수의 형도 배정확도형으로 전환되며 결과도 배정확도형으로 된다. 만약 연산수들이 옹근수형이면 결과도 옹근수형이다. 여기서 중요한것은 연산수자체의 형은 바뀌지 않는다는것이다.

실례:

```
$foo = "0";  // $foo는 0이다.(ASCII 48)
$foo++;      // $foo는 1이다.(ASCII 49)
$foo += 1;   // $foo는 현재 2이다.
$foo = $foo + 1.3;  // $foo는 3.3으로 된다.
```

우의 실례결과를 실지로 확정해보려면 우의 코드를 아래의 코드에 삽입하면 된다.

```
echo "\$foo==$foo; 형 is " . get( $foo ) . "<br>\n";
```

주의: 배렬형에서 자동형전환은 되지 않는다.

실례:

```
$a = 1;         // $a는 1이다.
$a[0] = "f";  //$a는 첨수가 0이고 값이 《f》 인 배렬로 된다.
```

실례:

```
$a = "1";       // $a는 문자 《1》이다.
$a[0] = "f";  // 이때에는 오유통보문이 발생한다.
```

PHP에서는 문자렬의 문자에 첨수를 통해 접근하는것이 가능하므로 우의 례제는 $a의 문자렬의 첫번째문자를 《f》로 하라는것인지 또는 $a를 《f》라는 문자렬을 첫번째원소로 하는 배렬로 만들라는것인가를 알수 없다.

이런 리유로 PHP 3.0.12와 PHP 4.0b3-RC4에서는 배렬에서의 자동형전환은 정의되여있지 않다.

6.2.3. 형변환(type casting)

PHP에서의 형변환은 C에서와 같다. 변환하려는 변수앞에 원하는 형이름을 괄호안에 넣어 써주면 된다.

다음과 같은 변환이 가능하다.

- (int), (integer); 옹근수형으로 변환
- (real), (double), (float); 실수형으로 변환
- (string); 문자렬형으로 변환
- (array); 배렬형으로 변환
- (object); 객체형으로 변환

실례:

```
$foo = 10;
$bar = (double) $foo;
```

특정한 형들사이의 형변환은 그 결과를 명확하게 알수 없는 경우가 있다. 스칼라값이나 문자렬을 배렬로 형변환하면 그 값은 해당 배렬의 첫번째 원소의 값이 된다.

실례:

```
$var = 'ciao';
$arr = (array) $var;
echo $arr[0];
```

스칼라값이나 문자렬을 객체로 형변환하면 그 값은 해당 객체의 《scalar》라는 속성값이 된다.

실례:

```
$var = 'ciao';
$obj = (object) $var;
echo $obj->scalar;  // 출력값은 《ciao》이다.
```

6.2.4. 변수의 사용범위(Variable Scope)

변수는 그 변수가 선언된 범위내에서만 사용가능하다. PHP변수의 대부분은 하나의 범위(single scope)만을 가지고있다. 이 범위는 include나 require되는 파일에도 동일하게 적용된다.

실례:

```
$a = 1;
include "b.inc";
```

우에서 $a변수는 b.inc스크립트에서도 사용할수 있다.

기본적으로 함수안에서 선언된것은 함수안에서만 사용되는 국부변수이다.

실례:

```
$a = 1; /* 대역변수*/
   Function Test () {
     echo $a; /* 국부변수로 참조한다.*/
}

Test ();
```

이 실례에서는 아무런 값도 출력되지 않는다. 이것은 C와는 약간 차이가 있다.

PHP에서 대역변수를 사용하는 첫번째방법은 대역변수를 함수안에서 global으로 선언해주는것이다.

실례:

```
$a = 1;
$b = 2;
Function Sum () {
    global $a, $b;
      $b = $a + $b;
}
Sum ();
echo $b;
```

우의 실례는 《3》을 출력한다. 그것은 함수안에서 $a과 $b를 global로 선언했기때문이다. 함수안에서 리용할수 있는 대역변수의 개수에는 제한이 없다.

대역변수를 참조하는 두번째방법은 PHP에서 특별히 정의하고있는 $GLOBALS라는 배렬을 사용하는것이다. 우의 례를 다음과 같이 바꿔쓸수 있다.

실례:

```
$a = 1;
$b = 2;
 Function Sum () {
  $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}
 Sum ();
echo $b;
```

$GLOBALS배렬은 변수이름이 Key가 되고 그 변수의 내용이 값이 되는 원소를

가진 련상(associative)배렬이다.

변수에서 중요한것은 또한 정적변수(static variable)이다. 정적변수는 대역함수안에서만 존재하지만 대역함수가 완료되여도 그 값은 없어지지 않고 보존된다.

실례:

```
Function Test () {
        $a = 0;
        echo $a;
        $a++;
       }
```

우의 실례는 함수 Test()를 호출할 때마다 $a가 0으로 대입되므로 0을 출력한다.

실례:

```
Function Test () {
        static $a = 0;
        echo $a;
        $a++;
 }
```

우의 실례는 함수 Test()를 호출할 때마다 처음에는 0, 다음에는 1, 그 다음에는 2, 이런식으로 하나씩 증가된 수를 출력해준다. 즉 $a의 값이 보존되는것이다.

또한 정적변수는 재귀적방법에서 반드시 필요하다. 재귀함수는 자기자체를 호출하는 함수를 말한다. 재귀적인 함수를 사용할 때는 무한고리에 빠질수 있기때문에 주의를 돌려야 한다. 따라서 재귀호출을 끝내는 적절한 방법을 적용해야 한다.

다음의 실례는 재귀적호출을 10번 진행한다.

실례:

```
Function Test () {
     static $count = 0;
     $count++;
     echo $count;
     if ($count < 10) {
        Test ();
     }
     $count--;
 }
```

6.2.5. 가변변수

때때로 변수의 이름을 변경할 필요가 제기된다. 여기에서는 변수이름을 바꾸는

방법에 대하여 보기로 하자.

보통의 변수선언은 다음과 같다.

```
$a = "hello";
```

여기서 《hello》는 변수의 값이다. 이 값을 변수이름으로 바꾸기 위해서는 아래와 같이 하면 된다.

실례:

```
$$a = "world";
```

이렇게 하면 《hello》라는 값을 가지는 변수 $a와 《world》라는 값을 가지는 변수 $hello가 생기게 된다. 그러므로 다음과 같은 두개의 코드

```
echo "$a ${$a}";
echo "$a $hello";
```

을 실행하면 둘다 《hello world》라는 결과를 얻는다.

가변변수를 배렬과 함께 사용하려면 모호성문제를 해결해야 한다. 그것은 만일 $$a[1]이라고 썼으면 $a[1]을 하나의 변수로 볼것인가 아니면 $$a를 변수로 보고 [1]를 그 변수의 첨수로 볼것인가 하는 모한 문제가 제기된다. 여기서 전자의 경우 ${$a[1]}라고 쓰고 후자의 경우는 ${$a}[1]라고 쓰면 우의 모호성문제도 해결된다.

가변변수를 표현할 때에는 항상 중괄호({ })를 리용한다.

6.2.6. PHP 태그밖에서 정의된 변수들

- HTML대화칸

홈대화칸이 PHP스크립트로 넘어오게 되면 대화칸에 있는 모든 내용들이 자동적으로 만들어진 PHP변수로 들어온다.

실례:

```
<form action="foo.php3" method="post">
    이름: <input type="text" name="이름"><br>
    <input type="submit">
</form>
```

실례의 대화칸이 넘어오게 되면 PHP는《$이름》형식의 변수를 만들고 이 변수에 대화칸의 이름, 마당에 입력된 모든 내용을 저장한다.

PHP는 대화칸에 1차원배렬변수도 사용할수 있다. 례를 들어 여러 변수를 함께 사용하는 그룹관련변수나 다중선택변수들의 내용들을 검색할수 있다.

실례:

```
<form action="array.php" method="post">
    이름: <input type="text" name="personal[이름]"><br>
    Email: <input type="text" name="personal[email]"><br>
```

```
        Beer: <br>
        <select multiple 이름="beer[]">
            <option value="warthog">Warthog
            <option value="guinness">Guinness
            <option value="stuttgarter">Stuttgarter
         </select>
        <input type="submit">
    </form>
```

만약 php.ini파일에서 track_vars이 설정되여 있다면 POST나 GET방식으로 전송되는 모든 변수들과 그 내용은 대역배렬변수인 $HTTP_POST_VARS과 $HTTP_GET_VARS에서 찾을수 있다.

-화상단추의 추가

홈대화칸에서 어떤 페지로 자료를 보낼 때 일반적으로 보내기단추(submit)대신 아래와 같이 그림을 사용할수도 있다.

실례:

```
<input type=image src="image.gif" 이름="sub">
```

사용자가 화상을 찰칵하면 대화칸에는 sub_x와 sub_y의 두개 변수가 추가되여 봉사기에 전송된다. 이 두 변수는 그림에서 사용자가 찰칵한 위치의 정보를 담고있다.

일부 열람기에서는 밑줄(_)대신 점(.)을 사용하는 경우도 있는데 PHP는 이런 경우 자동적으로 점(.)을 밑줄(_)로 바꿔준다.

- HTTP쿠키

쿠키는 봉사기가 사용자식별 등을 위해 의뢰기자료를 해당 의뢰기에 저장한 자료를 말한다. 쿠키를 작성하기 위해 SetCookie()함수를 사용한다.

쿠키는 HTTP머리부의 한 부분이므로 SetCookie()함수는 열람기로 보내는 어떤 자료내용보다도 먼저 서술해야 한다. 이것은 Header()함수와 같은것으로 생각하면 된다.

사용자가 보내준 모든 쿠키는 자동적으로 PHP변수로 변환된다. 만일 동일한 쿠키에 여러값을 저장하고 싶다면 쿠키이름에 《[]》를 추가하면 된다.

실례:

```
SetCookie ("MyCookie[]", "Testing", time()+3600);
```

쿠키를 새로 만들면 경로나 령역이 다르지 않는 한 이전의 쿠키를 덧쓰게 된다.

실례:

```
$Count++;
SetCookie ("Count", $Count, time()+3600);
```

SetCookie ("Cart[$Count]", $item, time()+3600);

- 환경변수(Environment variables)

PHP는 자동적으로 환경변수들을 일반적인 PHP변수로 만든다. GET, POST나 Cookie를 통해서 정보가 들어오면 그것으로부터 자동적으로 PHP변수가 생기므로 환경변수들을 읽어들여 정확한 판본을 확인하려는 경우에 좋다. 이를 위해 getenv()함수를 사용할수도 있으며 환경변수를 설정하기 위해 putenv()함수를 사용할수도 있다.

- 외부변수명에 있는 《.》의 처리

일반적으로 PHP는 변수를 스크립트로 가져올 때 변수명을 바꾸지 않는다. 그러나 아래와 같이 PHP의 변수명에 점이 있는 경우에는 변수명을 바꾼다.

$var이름.ext; /*무효한 변수이름 */

우의 경우 구문해석기는 《$var이름》이라는 변수와 련결연산자, 'ext'(인용부호가 없는 경우 그것이 어떤 알려진 열쇠어나 예약어가 아니면 문자렬로 취급된다.)로 해석한다. 이것은 원하는 결과가 아니다.

우와 같은 리유로 PHP는 외부에서 불러들인 변수의 이름에 《.》이 있으면 《_》(밑줄)로 바꾸어 읽어들인다.

- 변수의 형판단

PHP는 변수의 형을 자체로 정하고 필요한 경우에는 그 형을 바꾸므로 특정한 시점에서 그 변수가 어떤 형인지는 쉽게 알수 없다. PHP는 변수의 형을 알아보기 위한 여러개의 함수를 가지고있다. 즉 그 함수에는

get(), is_long(), is_double(), is_string(), is_array(), and is_object()

들이 있다.

6.2.7. 상수

PHP에서는 기본 상수들을 제공하고 실행시 상수를 설정할수 있도록 지원한다. 상수는 변수와 비슷하다. 그러나 상수는 define()함수를 사용하여 선언한다는것과 코드작성과정에 다른 값으로 바꿀수 없다는 점에서 변수와 차이난다.

- 미리 지정된 상수

- __FILE__

현재 처리중인 스크립트의 파일이름이다. 현재 파일이 include나 require를 리용하여 포함된 파일이라면 include함수 등을 호출한 부모파일이 아닌 그 파일의 이름을 가리킨다.

- __LINE__

스크립트내에서 현재 처리중인 명령의 행번호이다. 현재 파일이 include나 require로 포함된 파일이라면 include함수 등을 호출한 부모파일이 아닌 그 파일내에서의 행번호이다.

- PHP_VERSION

현재 사용중인 PHP구문해석기의 판본이다.(례: '3.0.8-dev')

- PHP_OS

PHP구문해석기가 실행되고있는 조작체계의 이름이다.(례: 'Linux')

- TRUE

참값

- FALSE

거짓값

- E_ERROR

문법오유가 아닌 회복이 불가능한 오유를 표시한다.

- E_WARNING

PHP가 오유를 발견했어도 실행은 계속 된다.

- E_PARSE

구문해석기가 스크립트파일에서 문법적으로 틀린 명령을 만난 경우이다. 회복은 불가능하다.

- E_NOTICE

오유는 아니지만 어떤 알려줄 사항이 있음을 나타낸다. 실행은 계속된다. 례를 들어 hash첨수에 인용부호가 없는 문자렬이 사용되였거나 이전에 선언되지 않은 변수의 값을 읽은 경우이다.

E_* 상수는 대체로 error_reporting()함수를 사용하여보고 준위를 정할 때 흔히 사용된다.

또한 define()함수를 사용하여 새로운 상수를 정의할수 있다.

실례1:

```
<?php
  define("CONSTANT", "Hello world.");
  echo CONSTANT; // "Hello world."를 출력
?>
```

실례2:

```
<?php
   function report_error($file, $line, $message) {
     echo "An error occured in $file on line $line:
         $message.";
   }
     report_error(__FILE__,__LINE__, "Something went
         wrong!");
```

```
?>
```

제3절. 표현식과 연산자

6.3.1. 표현식

표현식은 PHP에서 매우 중요하다. PHP에서 작성하는 거의 모든것은 다 표현식이다. 실례로 《$a = 5》를 들수 있다. 이것은 5라는 값을 $a라는 변수에 대입하는 표현식이다.

PHP는 정수값(integer values), 실수값(floating point values), 문자렬(string values)의 3가지 스칼라형값을 가진다.(스칼라값이란 더 이상 쪼개질수 없는 작은 값을 말한다.) 또한 PHP는 배렬(array)과 객체(object)의 2가지 혼합형을 지원한다. 이 형들은 변수에 값을 할당할수 있고 함수에 의해 값을 되돌려줄수 있다.

PHP는 거의 모든것이 표현식으로 표현된다는 점에서 표현중심언어 (expression-oriented language)라고 부른다.

표현중심의 대표적인 실례는 대입문이다. PHP에서는 C와 같이 대입문도 하나의 표현식으로 본다. 따라서 《$a=5; $b=5;》는 《$a = ($b=5);》 혹은 《$a = $b = 5;》으로도 표현할수 있다.

표현중심의 다른 례로 ++와 --를 들수 있다.

PHP/FI2에서 《$a++》와 같은것은 아무 값도 가지지 않는다. 즉 표현식이 아니다. 그러므로 이 값을 다른데 대입하는것은 불가능하다.

그러나 PHP3에서는 이것이 가능하다. C에서와 같이 ++/--를 사용하는 방법에는 앞에 붙이는것과 뒤에 붙이는것 두가지가 있다. 둘다 변수의 값이 증가된다는 점은 같으나 《++$variable》은 그 변수의 증가된 값이 표현식의 값으로 되고 《$variable++》는 증가되기 전의 변수값이 표현식의 값으로 된다.

비교표현식은 0이나 1의 값을 가진다. 0은 거짓(FALSE)을, 1은 참(TRUE)을 나타낸다.

PHP는 >(보다 크기), >= (보다 크거나 같기), == (같기), < (보다 작기), <= (보다 작거나 같기)를 지원한다.

마지막 례는 결합된 연산자-대입 표현식(combined operator-assignment expression)이다. 《$a = $a + 3》은 《$a += 3》으로 표현할수 있다. -=, *=, /=, .= 등도 사용할수 있다.

실례:

```
function double($i) {
        return $i*2;
    }
$b = $a = 5;  /* $a와 $b 에 5 대입 */
```

```
$c = $a++;    /*$c에는 $a의 원래 값인 5가 대입되고 $a는 6이 된다 */
$e = $d = ++$b; /*$d와 $e는 증가된 $b의 값 6이 대입된다. */
$f = double($d++); /* $f는 $d가 증가되기 전의 값인 6을 두 배한 값을 가진다. 2*6 = 12 */
$g = double($++e); /* $g는 $e가 증가된 후의 값인 7을 두배한 값을 가진다. 2*7 = 14 */
$h = $g += 10; /* $g는 14에 10을 더해 24의 값을 가진 후 그 값을 $h에 대입한다. 둘다 24 */
```

그러나 하나의 표현식이 하나의 문장은 아니다. 표현식이 반두점(;)으로 끝나야 하나의 문장이 되는것이다. 즉 《$b=$a=5;》에서 $a=5는 표현식이지만 문장은 아니다.

PHP에서 참/거짓의 판단은 Perl과 비슷하다. 0이 아닌 수자는 모두 TRUE이고 0은 FALSE이다. 빈 문자렬이나 문자렬 《0》은 FALSE이고 다른 모든 문자렬은 TRUE이다. 배렬이나 객체에 원소가 하나도 없다면 FALSE를 의미하고 나머지의 경우는 TRUE이다.

6.3.2. 연산자

- 산수연산자

표 6-2. 산수연산자

기호	이 름
+	더하기
-	덜기
*	곱하기
/	나누기
%	나머지

- 대입연산자

기본대입연산자는 《=》이다. 이 연산자의 의미는 《왼변과 오른변이 같다》라는 뜻이 아니라 오른쪽의 표현식을 계산하여 그 값을 왼쪽에 있는 연산수의 값으로 설정한다는 의미이다.

대입연산자의 값은 왼쪽에 대입된 값이다. 즉 《$a = 3》의 값은 3이 된다.

실례:

```
$a = ($b = 4) + 5; // $a는 9, $b는 4의 값을 가진다.
```

기본대입연산자외에 비트연산자, 산수연산자와 결합한 복합대입연산자도 있다.

실례:

```
$a = 3;
$a += 5; // $a는 8이다. 《$a = $a + 5;》 와 동일하다.
$b = "Hello ";
$b .= "There!"; // $b는 《Hello There!》로 된다.
                  《$b = $b . "There!";》와 같다.
```

- 비트연산자

비트연산자는 비트별로 론리연산을 진행한다.

표 6-3. 비트연산자

기호	이름
&	비트론리적
|	비트론리합
^	배타적론리합
~	비트반전

<<	왼쪽옮김
>>	오른쪽옮김

-비교연산자

비교연산자는 두개의 값을 비교하는 연산자이다.

표 6-4. 비교연산자

기호	이름
==	같기
===	같기(형도 같기)
!=	같지 않기
<	작기
>	크기
<=	작거나 같기
>=	크거나 같기

그밖에 조건연산자로서 《?:》이 있다.

형식은 다음과 같다.

```
(expr1) ? (expr2) : (expr3);
```

이 표현식은 expr1의 결과가 참이면 expr2를 반환하고 거짓이면 expr3을 반환한다.

-오유조종연산자

PHP는 《@》라는 한개의 오유조종연산자를 제공한다. PHP의 표현식앞에 이 표시가 붙으면 해당 표현식에서 발생한 모든 오유통보문이 무시된다.

만약 php.ini파일에서 track_errors기능이 설정되여있다면 해당 표현식에서 발생한 모든 오유통보문은 $php_errormsg라는 전역변수에 저장된다. 이 변수의 값은 매번 오유가 발생할 때마다 새로 설정된다. 따라서 만약 이 변수를 사용하려면 오유가 발생한 직후에 최대한 빨리 사용하여야 한다.

실례:

```
<?php
        /* SQL오유의 발생시 $php_errormsg변수의 리용 */
       $res = @mysql_query( "select이름, code from '이름list" ) or
      die( "질문오유발생: 오유는 다음과 같다.'$php_errormsg'" );
?>
```

- 실행연산자

PHP는 실행연산자를 제공한다.

PHP는 이 연산자사이에 있는 명령을 shell명령어로 실행하고 그 결과를 반환한다. 실행결과를 단순히 출력하는것뿐만 아니라 변환하여 변수에 저장할수도 있다.

실례:

```
$output = `ls -al`;
    echo "<pre>$output</pre>";
```

- 증가/감소 연산자

PHP는 C언어에서처럼 앞, 뒤에 붙는 증가, 감소 연산자를 제공한다.

표 6-5. 증가/감소연산자

기 호	의 미
++$a	먼저 1만큼 증가시키고 반환
$a++	먼저 반환한 후 1만큼 증가
--$a	먼저 1만큼 감소시키고 반환
$a--	먼저 반환한 후 1만큼 감소

실례:

```
<?php
echo "<h3>Postincrement</h3>";
$a = 5;
echo "Should be 5: " . $a++."<br>\n";
echo "Should be 6: " . $a."<br>\n";

echo "<h3>Preincrement</h3>";
$a = 5;
echo "Should be 6: " . ++$a . "<br>\n";
echo "Should be 6: " . $a . "<br>\n";

echo "<h3>Postdecrement</h3>";
$a = 5;
echo "Should be 5: " . $a-- . "<br>\n";
echo "Should be 4: " . $a . "<br>\n";
```

```
echo "<h3>Predecrement</h3>";
$a = 5;
echo "Should be 4: " . --$a . "<br>\n";
echo "Should be 4: " . $a . "<br>\n";
?>
```

- 론리연산자

론리연산은 론리형자료에 대하여 AND, OR연산을 진행하며 연산결과는 론리형이다.

표 6-6. 론리연산자

기호	이름
And	론리적
Or	론리합
Xor	배타적론리합
!	부정
&&	론리적

- 연산자우선순위(Precedence)

연산자들의 우선순위는 두개의 표현식중 어느것이 먼저 실행되는가를 나타낸다.

례를 들어 《1 + 5 * 3》은 결과가 18이 아니라 16이다. 그것은 곱하기연산자(*)가 더하기연산자(+)보다 우선순위가 높기때문이다.

아래에서는 연산자의 우선순위를 나타낸다. 여기서는 아래로 내려가면서 연산자의 우선순위가 낮아진다. 즉 맨 마지막 연산자의 우선순위가 제일 낮다.

```
.
or
xor
and
print
=, +=, -=, *=, /=, .=, %=, &=, |=, ^=, ~=, <<=, >>=
?, :
||
&&
|
^
```

&

==, !=, ===

<, <=, >, >=

<<, >>

+, -, .

*,/,%

! ,~ ,++, --, (int), (double), (string) ,(array) ,(object), @

[

new

- 문자렬연산자

문자렬연산자에는 두개가 있다. 하나는 왼쪽과 오른쪽의 두 문자렬을 련결하는 련결연산자(.)이고 다른 하나는 련결대입연산자(.=)이다.

실례:

```
$a = "Hello ";
$b = $a ."World!"; //현재 $b는 "Hello World!"이다.
$a = "Hello ";
$a .= "World!"; // 현재 $a는 "Hello World!"이다.
```

제4절. 조종구조

PHP는 몇개의 구문으로 이루어진다. 하나의 구문은 대입문이 될수도 있고 함수호출문, 반복문, 조건문이 될수도 있다. 한 구문은 일반적으로 반두점(;)으로 끝난다. 또한 여러개의 구문이 《{,}》를 리용하여 하나의 구문으로 되여 사용될수도 있다.

여기서는 여러가지 구문의 형태에 대해 보기로 한다.

- if문

PHP의 if문은 C에서와 비슷하다.

형식은 다음과 같다.

```
if (expr)
    statement
```

expr을 평가하여 TRUE이면 statement를 실행한다. FALSE이면 무시한다.

실례:

```
if ($a > $b)
```

```
print "a is bigger than b";
```

하나의 if문에서 여러 구문을 리용하려면 중괄호({ })를 사용하여 복합문으로 만들면 된다.

실례:

```
if ($a > $b) {
    print "a is bigger than b";
    $b = $a;
}
```

-else문

else는 if문의 평가식이 false일 때 실행하여야 할 구문을 지정한다.

실례:

```
if ($a > $b) {
   print "a is bigger than b";
} else {
   print "a is not bigger than b";
}
```

-elseif문

elseif는 else와 if를 합쳐놓은것과 같다.

실례:

```
if ($a > $b) {
      print "a is bigger than b";
  } elseif ($a == $b) {
      print "a is equal to b";
  } else {
      print "a is smaller than b";
  }
```

하나의 if문에는 여러개의 elseif문이 있을수 있다.

어떤 elseif문이 실행되려면 if문의 평가식과 그 앞의 모든 elseif문의 평가식이 false이여야 한다.

- 조종구조의 다른 표현

PHP3에서는 { }를 쓰는 대신 if(expr)뒤에 두점(:)을 찍고 하나이상의 문장을 라렬한 후에 《endif;》로 끌낼수도 있다.

이 방법은 특히 if문안에 HTML구문을 삽입하려는 경우에 리용한다.

실례:

```
<?php if ($a == 5): ?>
  A is equal to 5
<?php endif; ?>
```

우의 실례에서 "A = 5"라는 HTML구문이 if문안에서 사용되고있다.
다음과 같이 else와 elseif(expr)에서도 그것을 사용할수 있다.

실례:

```
if ($a == 5):
        print "a is equal to 5";
        print "...";
elseif ($a == 6):
        print "a is equal to 6";
        print "!!!";
else:
        print "a is neither 5 nor 6";
endif;
```

-while문

while문은 PHP3의 가장 간단한 형태이다.

형식1:

```
while (expr) statement
```

f문과 마찬가지로 중괄호({ })를 사용하지 않는 방법도 있다.

형식2:

```
while (expr): statement ... endwhile;
```

다음 두개의 실례는 1부터 10까지 출력하는 실례이다.

실례:

```
$i = 1;
  while ($i <= 10) {
      print $i++;  /* 증가되기전의 값인 1이
                      출력된다.(post-increment) */
  }
```

실례:

```
$i = 1;
```

```
while ($i <= 10):
    print $i;
    $i++;
endwhile;
```

- do…while문

do…while문은 비교식이 앞에 있는것이 아니라 맨 뒤에 있다는 점을 제외하면 while문과 비슷하다. 여기서 while조건문은 do문장이 실행된 후에 평가되므로 do이후의 문장은 무조건 한번은 실행된다.

실례:

```
$i = 0;
      do {
            print $i;
      } while ($i>0);
```

- for문

for문은 가장 많이 사용되는 명령문이다. for문의 형식은 다음과 같다.

for (expr1; expr2; expr3) statement

첫번째 표현식(expr1)은 순환을 진행할 때 무조건 한번 평가(실행)된다.

매 순환의 시작때마다 expr2가 평가된다. 만약 이것이 true면 순환은 계속되고 명령문(statement)이 실행된다. expr2가 false이면 순환은 완료된다.

매 순환이 끝날 때마다 expr3이 실행된다.

3개의 표현식이 없을수도 있지만 expr2가 없으면 무한순환이 진행된다. 이때 break명령문을 리용하여 완료할수 있다.

다음 실례들은 1부터 10까지 출력하는 실례로서 여러가지 방법으로 실현할수 있다는것을 보여준다.

실례:

```
for ($i = 1; $i <= 10; $i++) {
    print $i;
}
```

실례:

```
for ($i = 1;;$i++) {
   if ($i > 10) {
      break;
   }
      print $i;
```

```
}
```

실례:

```
$i = 1;
for (;;) {
if ($i > 10) {
      break;
}
      print $i;
      $i++;
 }
```

실례:

```
for ($i = 1; $i <= 10; $i++) print $i ;
```

PHP는 for문에 대해서도 《:》를 지원한다.

형식은 다음과 같다.

```
for (expr1; expr2; expr3): statement; ...; endfor;
```

- foreach문

PHP4는 perl 등의 다른 언어에서 리용하는 foreach문도 제공한다.(PHP3은 이 구조를 제공하지 않는다.)

이 구조는 배렬에서 반복적인 작업을 하는데 유용하다. 이것에는 두가지 형식이 있다.

형식은 다음과 같다.

```
foreach(array_expression as $value) statement
foreach(array_expression as $key => $value) statement
```

첫번째 형식은 array_expression으로 주어진 배렬에 대해 순환을 진행한다. 매번 순환할 때마다 배렬의 원소는 $value에 저장되고 내부배렬지적자(internal array pointer)는 하나 증가하여 다음 순환을 진행할 때 새로운 원소를 참조하도록 한다. 두번째 형식은 첫번째와 동일한 작업을 하지만 $key에 해당 원소의 키값을 저장한다.

foreach문이 처음 수행될 때 내부배렬지시문(internal array pointer)은 자동적으로 배렬의 첫번째 원소로 설정된다. 그러므로 foreach문을 사용할 때 reset()를 미리 호출할 필요가 없게 된다.

이 형식들을 리용한 아래의 2개 실례는 실행하면 동일한 결과가 얻어진다.

실레:

```
reset ($arr);
while (list(, $value) = each ($arr)) {
    echo "Value: $value<br>\n";
}
foreach ($arr as $value) {
    echo "Value: $value<br>\n";
}
```

실례:

두번째 형식으로 우의 실례를 표현하면 다음과 같다.

```
reset ($arr);
while (list($key, $value) = each ($arr)) {
    echo "Key: $key; Value: $value<br>\n";
}

foreach ($arr as $key => $value) {
    echo "Key: $key; Value: $value<br>\n";
}
```

추가적으로 몇개의 실례를 더 보자.

실례:

```
$a = array (1, 2, 3, 17);
   foreach ($a as $v) {
   print "Current value of \$a: “.$v.” \n";
}
```

실례:

```
$a = array (1, 2, 3, 17);
$i = 0;
foreach($a as $v) {
    print "\$a[$i] => $k \n";
}
```

실례:

```
$a = array ("one" => 1,"two" => 2,"three" => 3,
    "seventeen" => 17
);
```

```
foreach($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}
```

- break문

break문은 현재 순환에서 빠져나가는 명령이다.

break문에는 수자선택항목을 줄수 있는데 이것은 한번에 빠져나갈 조종구조의 수를 의미한다.

실례:

```
$i = 0;
  while ($i < 10) {
      if ($arr[$i] == "stop") {
          break;
      }
      $i++;
  }

$i = 0;
while ( ++$i ) {
    switch ( $i ) {
    case 5:
        echo "At 5<br>\n";
        break 1;
   case 10:
        echo "At 10; quitting<br>\n";
        break 2;
    default:
        break;
    }
}
```

-continue문

continue문은 현재 고리의 시작점으로 가도록 하는 명령이다.

continue문도 수자선택항목을 줄수 있는데 이것은 한번에 시작점으로 갈 조종구조의 수를 의미한다.

실례:

```
while (list ($key, $value) = each ($arr)) {
    if (!($key % 2)) {
        continue;
    }
    do_something_odd ($value);
}
$i = 0;
while ($i++ < 5) {
        echo "Outer<br>\n";
        while (1) {
        echo "Middle<br>\n";
        while (1) {
        echo "Inner<br>\n";
        continue 3;
        echo "This never gets output.<br>\n";

        echo "Neither does this.<br>\n";
}
```

-switch문

switch문에 있는 평가식($i)과 일치하는 case문을 찾아 그 다음의 switch문이 끝날 때까지 실행한다. 평가식과 일치하는 case문이 없는 경우에는 무환순환에 빠질수 있으므로 break문을 리용한다.

실례:

```
switch ($i) {
    case 0:
        print "i equals 0";
    case 1:
        print "i equals 1";
    case 2:
        print "i equals 2";
    break;
}
```

특별한 case문으로는 default case가 있다. 이것은 평가식과 일치하는 case문이

하나도 없을 때 여기의 명령문을 실행한다.

실례:

```
switch ($i) {
  case 0:
      print "i equals 0";
      break;
  case 1:
      print "i equals 1";
      break;
  case 2:
     print "i equals 2";
     break;
 default:
     print "i is not equal to 0, 1 or 2";
}
```

case평가식으로는 옹근수, 실수, 문자렬, 스칼라형의 어떤 표현식도 된다. 배렬이나 객체도 문제로 될것은 없지만 그것은 문장에서 의미가 없다.

-require문

require문은 C언어의 #include와 비슷하게 자신을 지정한 파일로 교체한다.

include()되거나 require()되여 포함되는 파일은 처음에 PHP방식에서 빠져나와 HTML방식으로 들어가고 마지막에 PHP방식으로 복귀한다. 따라서 포함될 파일의 PHP코드는 적절한 PHP시작, 완료꼬리표에 둘러싸여 있어야 한다.

require()문은 고리구조안에 넣고 매번 다른 파일을 읽어들일수 없다. 이런 경우에는 include()문을 사용하여야 한다.

실례:

```
require('header.inc');
```

require()는 사실 함수가 아니라 조종구조이다. 그러므로 이것은 함수와는 다른 규칙에 따른다. 또한 require()는 다른 어떤 조종구조와 함께 사용될수 없고 되돌림값도 없다.(되돌림값을 받으려 한다면 구문해석오유가 발생한다.)

include()와 다르게 require()는 언제나 해당 파일을 읽어온다. 심지어 해당 행이 전혀 실행되지 않아도 읽어온다. 만약 조건에 따라 파일을 포함시키고싶다면 include()문을 사용하여야 한다. 조건문은 require()에 아무 영향을 미치지 못한다. 그러나 require()문이 있는 행이 실행되지 않는다면 해당 파일의 어떤 코드도 실행되지 않으며 순환문 또한 require()에 영향을 주지 못한다. 해당 파일에 있는 내용이 고리의

내용이라 하더라도 require()문은 단지 한번만 나타나는것으로 된다.

이것은 require()문을 순환문안에서 사용할수 없다는것을 의미하고 매 순환시마다 다른 파일을 읽어오려면 include()문을 사용하여야 한다는것이다.

include()와 require()는 모두 호출한 스크립트안으로 원하는 파일의 내용자체를 끌어들이는것이지 HTTP나 그와 비슷한 방식으로 해당 목표를 불러들이는것은 아니다. 따라서 현재 범위안에 선언된 모든 변수들은 포함할 파일안에서도 그대로 사용된다.

실례:

```
require ("file.inc?varone=1&vartwo=2"); /* Won't work. */
$varone = 1;
$vartwo = 2;
require ("file.inc");
```

remote files기능을 사용하여 HTTP를 통한 원격파일을 include()나 require()를 리용할 때 우의 내용을 정확히 알아둘 필요가 있다.

PHP3에서는 require()로 포함된 파일안에서 return문을 사용할수 있다. 그러나 return문이 포함된 파일의 대역적범위에서 나타나는 경우에만 가능하고 그 어떤 블로크내({} 내부)에서 사용할수 없다. 그러나 PHP4에서는 이런 기능자체가 없어졌다. 만약 이런 기능을 사용하고싶다면 include()문을 사용하면 된다.

- include문

include문은 지정한 파일을 읽고 실행한다.

include()나 require()로 읽어지는 파일은 포함된 파일의 처음에 PHP방식에서 빠져나와 HTML방식로 들어가고 마지막에 PHP방식로 복귀한다. 따라서 포함될 파일의 PHP코드는 적절한 PHP시작, 완료 꼬리표에 둘러싸여있어야 한다.

이 동작은 실행중에 include()문을 만날 때마다 일어난다. 따라서 include()문을 고리구조안에 두고 매번 다른 파일을 읽어들이도록 한다.

실례:

```
$files = array ('first.inc', 'second.inc', 'third.inc');
for ($i = 0; $i < count($files); $i++) {
        include $files[$i];
}
```

include()는 이 문장을 만날 때마다 매번 재평가되여 재실행된다는 점에서 require()와 다르다. 반면에 require()문은 지정된 파일의 내용이 실행되는가에 관계없이(례를들어 if문안에 들어있고 상태가 거짓인 경우에도) 이 문장을 처음 만났을 때 지정된 파일로 교체된다.

include()는 특별한 구조이므로 만약 이것이 조건부문안에 놓여있다면 반드시

{}(statement block)으로 둘러싸야 한다.

실례:

```
if ($condition) {
    include($file);
} else {
    include($other);
}
```

PHP3, PHP4에서 모두 include()된 파일내에서 이 파일의 수행을 완료하고 이 파일을 부른 스크립트로 복귀하기 위해서는 return문을 사용할수 있다. 그런데 약간의 차이가 있다.

우선 PHP3에서는 해당 블로크가 함수의 블로크가 아닌 이상 return 문이 블로크안에 올수 없다.(함수의 블로크안에 있는 경우는 해당 함수에서 return하는것이지 현재 파일에서 return하는것은 아니다.) 그러나 PHP4에서는 이 제한이 없다. 또한 PHP4에서는 include()파일의 return시에 되돌림값을 사용할수 있다. include()문을 일반 함수처럼 반환값을 받을수 있다.(이것은 PHP3에서는 구문오유를 발생시킨다.)

제5절. 함수와 클라스

6.5.1. 리용자정의함수

함수는 다음과 같이 정의한다.

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
    echo "Example function.\n";
     return $retval;
}
```

함수안에는 다른 함수나 class의 선언 등이 포함될수 있다.

PHP3에서 함수를 사용하기전에 우선 선언하여야 하였으나 PHP4에서는 이런 제한조건이 없다.

PHP는 함수의 다중정의(function overloading)를 지원하지 않으며 이미 정의된 함수를 없애지 못한다.

PHP3에서는 함수파라메터의 기정인수값(default argument values)을 설정해줄수 있으며 파라메터의 개수를 가변으로 설정하는것은 불가능하다. 그러나 PHP4는 두가지 모두 가능하다.

- 함수인수들

인수들을 통해 함수에 어떤 정보를 넘겨줄수 있다. 이 인수들은 반점(,)으로 구분되는 변수나 상수들이다.

PHP에서 함수인수들은 변수값, 참조변수, 기정인수의 3가지 형태로 리용한다.

가변길이(Variable-length)인수들은 PHP4이후에서만 제공된다. 그러나 PHP3에서는 배렬을 사용하여 가변길이인수를 표현할수 있다.

실례:

```
function takes_array($input) {
        echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

기본적으로 함수의 인수들은 값으로 전달된다. 함수내에서 변화시킨 값을 그대로 유지하려면 참조변수를 리용해야 한다.

어떤 함수의 인수를 항상 참조변수로 넘기려면 함수를 선언할 때 《&》기호를 인수의 앞에 붙여주면 된다.

실례:

```
function add_some_extra(&$string) {
            $string .= 'and something extra.';
    }
    $str = 'This is a string, ';
    add_some_extra($str);
    echo $str;
```

만약 변수값으로 호출하지 않고 참조변수로 호출하고싶다면 함수를 호출할 때 인수의 앞에 &를 붙이면 된다.

실례:

```
function foo ($bar) {
        $bar .= ' and something extra.';
}
$str = 'This is a string, ';
foo ($str);
echo $str;
foo (&$str);
echo $str;
```

-기정인수값들(Default argument values)

스칼라인수는 다음과 같이 C++와 비슷하게 기정값을 정해줄수 있다.

실례:

```
function makecoffee ($type = "cappucino") {
  return "Making a cup of $type.\n";
   }
echo makecoffee ();
echo makecoffee ("espresso");
```

우의 코드의 실행결과는 다음과 같다

```
Making a cup of cappucino.
Making a cup of espresso.
```

기정값은 반드시 상수여야 한다. 례를 들어 변수나 클라스의 성원을 사용해서는 안된다는것이다.

- 가변길이인수목록

PHP4에서는 사용자정의함수에 가변길이인수목록을 제공한다. func_num_args(), func_get_arg(), func_get_args()의 함수를 사용하여 쉽게 사용할수 있다.

특별한 문법이 사용되지도 않고 함수를 정의할 때나 사용할 때에 인수목록은 보통의 경우와 같이 사용하면 된다.

- 귀환값

함수는 return문을 통해 함수값을 돌려줄수 있다.

목록과 객체를 포함한 그 어떤 형도 돌려줄수 있다.

실례:

```
function square ($num) {
     return $num * $num;
}
echo square (4);   // '16'을 출력
```

여러개의 값을 되돌려주는것은 할수 없다. 그러나 목록를 되돌려줌으로써 비슷한 기능을 수행할수 있다.

실례:

```
function small_numbers() {
        return array (0, 1, 2);
}
        list ($zero, $one, $two) = small_numbers();
```

- old_function

old_function문장은 PHP/FI2에서 동일한 함수사용법을 제공한다.(function대신

old_function을 사용한다는 점은 제외) 이것을 사용하는것은 PHP/FI2->PHP3변환기에서뿐이다.

old_function으로 정의된 함수들은 PHP의 내부코드에서 호출될수 없다. 또한 usort()나 array_walk(), register_ shutdown_function()와 같은 함수에 사용할수 없다. 이것을 해결하기 위해서는 old_function으로 선언된 함수를 호출하는 PHP3형태의 함수를 만들어 사용한다.

- 가변함수

PHP는 가변함수(variable functions)를 지원한다. 이것은 변수이름뒤에 괄호가 왔을 때 PHP는 그 이름을 가진 함수를 찾아 실행한다는것을 의미한다. 이것을 callbacks, function table 등의 기능에 사용하면 매우 유용하게 사용할수 있다.

실례:

```
<?php
function foo() {
    echo "In foo()<br>\n";
}
function bar( $arg = '' ) {
    echo "In bar(); argument was '$arg'.<br>\n";
}

$func = 'foo';
$func();
$func = 'bar';
$func( 'test' );
?>
```

6.5.2. 클라스

클라스는 일련의 변수와 이 변수들을 사용하는 함수들의 모임이다. 클라스는 다음과 같은 형태로 선언된다.

```
<?php
class Cart {
var $items;
function add_item ($artnr, $num) {
    $this->items[$artnr] += $num;
}
function remove_item ($artnr, $num) {
```

```
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return true;
        } else {
            return false;
        }
    }
}
```

Cart라는 이름의 클라스인데 카트안에 들어있는 물품들을 의미하는 한개의 배렬 변수와 cart에 물건을 넣거나 빼는 두개의 함수로 구성되여있다.

new연산자를 사용하여 원하는 형변수를 생성하여야 한다.

실례:

```
$cart = new Cart;
$cart->add_item("10", 1);
```

우의 실례는 Cart클라스의 $cart라는 객체를 만드는 실례이다. 이 객체의 add_item()함수를 호출하여 물품번호가 10인 물품 1개를 카트에 넣는다.

클라스는 다른 클라스로 확장될수 있다. 확장 혹은 파생된 클라스는 기초로 되는 클라스의 모든 변수들과 함수들을 그대로 가지게 되고 여기에 추가로 확장된 선언을 할수 있다. 이를 위해 《extends》라는 열쇠단어가 사용된다.

실례:

```
class Named_Cart extends Cart {
    var $owner;
    function set_owner ($name) {
        $this->owner = $name;
    }
}
```

우의 례는 Cart클라스의 변수와 함수에 $owner변수와 set_owner()함수를 추가한 Named_Cart라는 클라스의 선언이다.

종래의 일반카트(normal cart)에 있던 함수도 사용할수 있다.

실례:

```
$ncart = new Named_Cart;
$ncart->set_owner ("kris");
print $ncart->owner;
$ncart->add_item ("10", 1);
```

클라스내부에 있는 함수에서 $this라는 변수는 자기자신 객체를 의미한다. $this->something의 형태로 현재 객체의 변수나 함수를 사용할수 있다. 구성자는 해당 클라스의 새로운 객체를 만들 때 자동적으로 실행되는 함수를 의미한다. 클라스의 이름과 같은 함수가 구성자가 된다.

실례:

```
class Auto_Cart extends Cart {
 function Auto_Cart () {
     $this->add_item ("10", 1);
 }
}
```

우의 실례는 Cart클라스에 물품번호 10번의 물품을 한개 자동으로 추가하는 구성자를 추가한 Auto_Cart라는 클라스의 선언이다. Auto_Cart는 새로 생성된다. 구성자는 또한 인수(argument)를 가질수 있고 이 인수들의 기정값을 선택항목으로 선언할수 있다. 구성자는 매우 유용하게 사용된다.

실례:

```
class Constructor_Cart extends Cart {
   function Constructor_Cart ($item = "10", $num = 1) {
       $this->add_item ($item, $num);
   }
}
$default_cart    = new Constructor_Cart;
$different_cart = new Constructor_Cart ("20", 17);
```

파생된 클라스의 경우에 이 클라스의 구성자가 호출될 때 부모클라스의 구성자는 자동으로 호출되지 않는다.

제6절. PHP의 응용

6.6.1. PHP의 환경설치

PHP를 Windows에 설치하기 위해서는 우선 다음의 프로그람원천들을 준비하여야 한다.

OS: Windows2000 Professional 이상

PHP: php-installer.exe

MySQL: mysql-win.zip

Apache: apache_win.exe

- MySQL설치하기
  - 먼저 mysql-win.zip를 풀고 setup.exe를 실행시킨다.
  - 설치가 끝났으면 다음과 같이 데몬(daemon)을 띄우고 실험해본다.

```
C:\>mysql\bin\mysqld-nt --install
C:\>Net start mysql
```

Mysql에서 빠져 나오려면

```
Mysql>quit
```

하면 된다. mysql에서 나왔다고 하여 데몬이 실행상태가 아닌것은 아니다.

그러므로 Mysql데몬을 끌내려면

```
C:>Net stop mysql
```

하면 된다.

- PHP설치하기
  - php-installer.exe을 실행시킨다.
  - C:/windows/php.ini파일을 열고

```
Doc_root= " C:\Program Files\ApacheGroup\Apache\htdocs\"
Extension_dir= "C:\php\"
```

로 수정한다.

- Apache설치하기
  - apache_win.exe을 실행시키면 Apache가 자동설치된다.
  - C:\ProgramFiles\ApacheGroup\Apache\conf\httpd.conf파일을 열고 ServerAdmin you@your.address부분을 찾아서 봉사기관리자 또는 개인이 설치하는 경우 자기 전자우편주소를 입력한다. 그리고 #ServerName new.host.name을 찾아서 그 아래에 ServerName http://localhost를 입력하고 보관한다.
  - C:\Program Files\ApacheGroup\Apache\conf에 있는 srm.conf파일을

열어서 하단에

```
ScriptAlias /php/ "C:/php/"
AddType application/x-httpd-php .phtml.html .htm .incAddType
application/x-httpd-php-source .phps
AddType application/x-httpd-php .php
AddType application/x-httpd-php .php3
Action application/x-httpd-php "/php/php.exe"
```

를 추가하고 보관한다.

- Apache를 재기동하면 설치가 끝난다.

Apache가 정확히 설치되였는가를 확인하려면 C:\Program Files\ApacheGroup\Apache\htdocs에서 index.html.en파일을 index.html로 이름을 바꾸고 열람기를 실행시켜 http://localhost를 입력한다.

- PHP가 정확히 설치되였는가를 확인해본다.

문서편집기인 Notepad로 PHP파일을 하나 작성한다.

실례로

```
<?php
phpinfo();
?>
```

로 된 info.php를 열람기로 실행시켜 phpinfo화면이 출력되면 정확히 설치되였다는것을 알수 있다.

6.6.2. 파일의 적재기능

웨브열람기를 리용하여 파일을 적재하는것은 전자게시판이나 전자우편을 실현하는데서 기초로 되는 문제이다. 파일을 적재한다는것은 사용자가 웨브열람기로 보내려는 파일을 선택하고 단추를 찰칵하면 그 파일이 웨브봉사기의 해당 등록부로 전송되도록 한다는것이며 파일의 적재는 웨브봉사에서 필연적으로 진행되는것이다.

그러면 파일의 적재가 어떤 과정을 통해 실현되는가를 보기로 하자

먼저 fileupload-class.php파일에 대하여 고찰해보자.

```
<?
class uploader{
 var $file;
 var $errors;
 var $accepted;
 var $max_filesize; //파일의 최대 바이트크기를 나타내는 변수
 var $max_image_width; //올리적재한 그림의 최대너비
```

```
var $max_image_height; //올리적재한 그림의 최대높이
//올리적재하려는 파일의 크기를 얻는 함수//
function max_filesize($size){
   $this->max_filesize = $size;
}
//올리적재하는 그림의 면적(너비,높이)을 얻는 함수(단위:px)
function max_image_size($width, $height){
   $this->max_image_width = $width;
   $this->max_image_height = $height;
 }
 //그 파일을 접수할수 있다면 그것을 선택하여 복사하는 함수
 function upload($filename='', $accept_type='', $extention='') {
 if (!$filename || $filename == "none") {
    $this->errors[0] = "No file was uploaded";
    $this->accepted = FALSE;
    return FALSE;
 }
 if($this->max_filesize && ($this->file["size"] > $this->max_filesize)) {
$this->errors[1] = "Maximum file size exceeded. File may be no larger than" .
$this->max_filesize/1000. "KB (" . $this->max_filesize . " bytes).";
 $this->accepted  = FALSE;
 return FALSE;
 }
if(ereg("image", $this->file["type"])) {
   $image = getimagesize($this->file["tmp_name"]);
   $this->file["width"] = $image[0];
   $this->file["height"] = $image[1];
if(($this->max_image_width||$this->max_image_height)&&
  (($this->file["width"]>$this->max_image_width)||  ($this->file["height"]
  > $this->max_image_height))) {
$this->errors[2] = "Maximum image size exceeded. Image may be no more
than " . $this->max_image_width . " x " . $this->max_image_height . "
pixels";
$this->accepted  = FALSE;
```

```
    return FALSE;
    }
        switch($image[2]) {
                case 1:
                        $this->file["extention"] = ".gif"; break;
                case 2:
                        $this->file["extention"] = ".jpg"; break;
                case 3:
                        $this->file["extention"] = ".png"; break;
                case 4:
                        $this->file["extention"] = ".swf"; break;
                case 5:
                        $this->file["extention"] = ".psd"; break;
                case 6:
                        $this->file["extention"] = ".bmp"; break;
                case 7:
                        $this->file["extention"] = ".tif"; break;
                case 8:
                        $this->file["extention"] = ".tif"; break;
                default:
                        $this->file["extention"] = $extention; break;
        }
  } elseif(!ereg("(\.)([a-z0-9]{3,5})$", $this->file["name"]) && !$extention)
  {
         switch($this->file["type"]) {
                case "text/plain":
                         $this->file["extention"] = ".txt"; break;
                case "text/richtext":
                         $this->file["extention"] = ".txt"; break;
                         default:
                         break;
         }
      } else {
          $this->file["extention"] = $extention;
```

```
        }
  if($accept_type) {
       if(ereg(strtolower($accept_type), strtolower($this->file["type"]))) {
         $this->accepted = TRUE;
       } else {
         $this->accepted = FALSE;
         $this->errors[3] = "Only " . ereg_replace("\|", " or ", $accept_type) .
                " files may be uploaded";
          }
         } else {
                $this->accepted = TRUE;
         }
         return $this->accepted;
}
  //파일이름을 정리하고 PHP의 temp위치에서 $path에로 파일을 복사하며
  //덮쓰기방식을 선택하는 함수
function save_file($path, $overwrite_mode="3"){
      $this->path = $path;
       if($this->accepted) {
       $this->file["name"] = ereg_replace("[^a-z0-9._]", "", str_replace("", "_",
str_replace("%20", "_", strtolower($this->file["name"]))));
if(ereg("text", $this->file["type"])) {
      $this->cleanup_text_file($this->file["tmp_name"]);
}
          if(ereg("(\.)([a-z0-9]{2,5})$", $this->file["name"])) {
               $pos = strrpos($this->file["name"], ".");
                if(!$this->file["extention"]) {
                     $this->file["extention"] = substr($this->file["name"],
                         $pos, strlen($this->file["name"]));
                  }
                 $this->file['raw_name'] = substr($this->file["name"], 0, $pos);
             } else {
                $this->file['raw_name'] = $this->file["name"];
                if ($this->file["extention"]) {
```

```
                $this->file["name"]=$this->file["name"].$this->file["extention
"];
                }
            }
          switch($overwrite_mode) {
           case 1: // 덧쓰기방식
            $aok=copy($this->file["tmp_name"],
$this->path.$this->file["name"]);
             break;
            case 2: //파일이름뒤에 증분량을 추가하는 식으로 새로 창조
             while(file_exists($this->path.$this->file['raw_name'].$copy.$th
is->file["extention"])) {
                $copy = "_copy" . $n;
                $n++;
                }
            $this->file["name"]=$this->file['raw_name'].$copy.$this->file["e
xtention"];
             $aok     =     copy($this->file["tmp_name"],     $this->path     .
$this->file["name"]);
                break;
             case 3: // 아무것도 하지 않는 방식( highest protection)
                if(file_exists($this->path . $this->file["name"])){
                 $this->errors[4]="File&quot".$this->path.$this->file["name"]
."&quot already exists";
                   $aok = null;
                } else {
                $aok=copy($this->file["tmp_name"],$this->path.$this->file["n
ame"]);
                }
                break;
                default:
                break;
                }
        if(!$aok) { unset($this->file['tmp_name']); }
```

```
                return $aok;
        } else {
                $this->errors[3]="Only   ".ereg_replace("\|",   "   or   ",
$accept_type)." files may be uploaded";
                return FALSE;
            }
        }

//MAC나  PC형식을 UNIX로 변환하는 함수
function cleanup_text_file($file){
          $new_file  = '';
          $old_file  = '';
          $fcontents = file($file);
         while (list ($line_num, $line) = each($fcontents)) {
              $old_file .= $line;
              $new_file .= str_replace(chr(13), chr(10), $line);
          }
         if ($old_file != $new_file) {
             // 올리적재된 파일을 열고 달라진 내용을 다시 쓰기
              $fp = fopen($file, "w");
              fwrite($fp, $new_file);
              fclose($fp);
         }
}
}
?>
```

upload.php파일을 보기로 하자

```
<html>
        <head><title>Upload</title></head>
<body>
<?php
require("fileupload-class.php");
$path = "uploads/";
$upload_file_name = "userfile";
```

```
// gifs형식만을 접수하는 경우
#$acceptable_file_types = "image/gifs";
// gif와 jpeg파일들을 접수하는 경우
#$acceptable_file_types = "image/gif|image/jpeg|image/pjpeg";
// 모든 파일들을 접수하는 경우
$acceptable_file_types = "";
//만일 주어진 확장자가 없는 경우 그리고 열람기나 PHP가 파일의 형태가 무엇인지
//분간하지 못한 경우 ".jpg"나 ".txt"와 같이 기정의 확장자를 추가할수 있다.
$default_extension = "";
// $path등록부에 같은 이름으로 된 다른 파일을 올리적재하려는 경우
//방식: 1 = 덧쓰기
//      2 = 증분량을 표시한 새로운  파일 생성
//      3 = 아무것도 하지 않음 (highest protection)
$mode = 2;
if ($PHP_REQUEST['submitted']) {
// 클라스의 새로운 객체를 생성
        $my_uploader = new uploader;
// 올리적재할수 있는 파일의 최대바이트크기를 설정
        $my_uploader->max_filesize(30000);
// 그림을 올리적재할 때 최대 px크기를 설정
        $my_uploader->max_image_size(800, 800); // max_image_size($width,
$height)
        //파일을 올리적재하기
        if ($my_uploader->upload($upload_file_name, $acceptable_file_types,
$default_extension)) {
            $success = $my_uploader->save_file($path, $mode);
        }
        if ($success) {
        // 파일의 올리적재가 성공
        print($my_uploader->file['name'] . " was successfully uploaded! <a
href=\"" . $_SERVER['PHP_SELF'] . "\">Try Again</a><br>");
        //print_r($my_uploader->file);와 같은 배렬을 모두 인쇄
        //혹은 파일을 인쇄
                if(ereg("image", $my_uploader->file['type'])) {
```

```
                echo "<img src=\"" . $path . $my_uploader->file['name'] .
"\" border=\"0\" alt=\"\">";
                } else {
                        $fp = fopen($path . $my_uploader->file['name'], "r
");
                        while(!feof($fp)) {
                                $line = fgets($fp, 255);
                                echo $line;
                                }
                        if ($fp) { fclose($fp); }
                        }
                } else {
                // 올리적재에서 오유발생
                        if($my_uploader->errors) {
                        while(list($key, $var)=each($my_uploader->errors))
    {
                                echo $var . "<br>";
                                }
                        }
                }
        }
?>
<form                                           enctype="multipart/form-data"
action="<?=$_SERVER['PHP_SELF']; ?>" method="POST">
        <input type="hidden" name="submitted" value="true">
                Upload this file:<br>
        <input name="<?= $upload_file_name; ?>" type="file">
                <br><br>
        <input type="submit" value="Send File">
        </form>
        <hr>
<?php
if ($acceptable_file_types) {
print("This form only acc+epts <b>" . str_replace("|","or",$acceptable_file_t
ypes)."</b> files\n");
        }
?>
</body>
</html>
```

찾 아 보 기